The Dissertation Committee for Shi Ying Lim Certifies that this is the approved version of the following dissertation:

Entrepreneurial Design of a Digital Health Business

Committee:

Sirkka L. Jarvenpaa, Supervisor

Edward G. Anderson Jr.

Diane Bailey

Douglas P. Hannah

Hüseyin Tanriverdi

Entrepreneurial Design of a Digital Health Business

by

Shi Ying Lim

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2018

Acknowledgements

I would like to begin by thanking my supervisor and mentor, Sirkka Jarvenpaa. You have been generous with your time, helped to read countless drafts, and is an inspiring role model for what it means to be an academic. To Hüseyin Tanriverdi, thank you for your sharp questions that constantly challenge me to be a better scholar. To Doug Hannah, thank you for the constructive and insightful advice, which has helped me tremendously in navigating the ups and downs of the latter years of my Ph.D. journey. To Ed Anderson, thank you for believing in me and my work. To Diane Bailey, thank you for the advice on research methodology and for providing a direction for this dissertation at a time when I was going around in circles.

My thanks also go to Jeffrey Treem. Your positivity always gave me hope and new insights when I felt dismal about my research. To the late Reuben McDaniel, Jr., thank you for imparting to me many life lessons. To Andy Whinston, thank you for prodding me away from studying health IT the very first semester of the program, and guiding me toward the field of entrepreneurship instead.

My thanks and appreciation also go out to the many faculty members, staff, and fellow students in the IROM and Management departments at McCombs who have supported and guided me on this humbling and arduous journey. To Ninja Janardhanan, Suho Han, and Laurie Giddens – thank you for your patience and willingness to listen to my incoherent ramblings, as I explored theory after theory. I will truly miss our conversations over coffee together. And to other friends with whom I have shared this journey: Abhishek Roy, Gorkem Ozer, Seung-Hwan Jeung, Hyerin Kim, Jordana George, Caroline Stratton, Christina Kyprianou, Shannon Provost, and the others I have gotten to

know through the McCombs doctoral community and conferences– thank you for your friendship.

My friends in Austin also deserve my most heartfelt thanks. I will always fondly remember Ella Segura, Audrey Kuang and Stephanie Cooley, with whom we spent many joyful hours during our time in Austin.

Lastly, to my family. To Melvin – We survived the Ph.D. journey together! I couldn't have done it without your unwavering love and support. To Claire and Brian – you are my biggest inspiration and cheerleaders. To my parents and siblings, thank you for always being there for me.

Entrepreneurial Design of a Digital Health Business

Shi Ying Lim, Ph.D.

The University of Texas at Austin, 2018

Supervisor: Sirkka L. Jarvenpaa

Software startups are an important source of innovation and wealth creation. Startups must develop and release software quickly to gain early cash flow and legitimacy for firm survival. They must search and identify a suitable market in a tight time frame. Aambiguity in the quality of the novel software designed can further intensify uncertainty for startups. Also, the entrepreneur-investor relationship is a critical conduit for financial and social resources, yet the investors' influence on software design is understudied. This leads to the following research question: How does the entrepreneurial context in which the startup operates affect software design process and product in the early years?

Given the exploratory nature of the research question, I drew on an eighteen-month, participant-observation case study of a three-year-old, digital healthcare startup, HealthCom. HealthCom designed a software to facilitate knowledge transfer between nurses and patients. I leveraged the attention-based view (ABV) as an organizing framework for analysis. By looking at how the environmental context influences the designers' attention and actions, researchers can begin to understand the rationale behind design decisions.

The findings illustrate how the attention of the designers were directed by financial-focused and quality-focused attention drivers. Financial-focused attention drivers originated from players (e.g. clients and investors) that intensified pressures for funding

and led to the reframing of software as an enabler of funding. This contradicted quality-focused design alternatives grounded in user requirements and software design rules. It underlined the move from attention to designs that accommodated user requirements towards those that captured emerging business opportunities. Designers had to balance the tensions between financial and quality-focused design decisions that could lead to inhibition of future growth, compromises in quality or over-optimization of quality. Designers simplified and coupled requirements throughout the design process and postponed investments of design resources to meet client-imposed deadlines. The design product became disposable, quick to build, limited in adaptability, spartan, and vulnerable, and it enabled the startup to capture the business opportunities amidst time constraints, financial pressures, and high uncertainty.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Software startups are an important source of innovation and new wealth creation (Gimmon & Levie, 2010; Grilli & Murtinu, 2012; Leong, Pan, Newell, & Cui, 2016; Nambisan, 2017), and are a growing phenomenon. Not only are software startups receiving more funding, but the average amount of money being invested in deals is larger than those in the dot.com bubble (Relander, 2015). Software startups need to manage both the design of the business model and the software product[1] in the early years (Antonopoulou, Nandhakumar, & Panourgias, 2014). Unfortunately, there are limited studies in Information Systems (IS) that provide insight into the way software design occurs in startups, particularly in the early years (Tumbas, Seidel, Berente & Brocke, 2015; Antonopoulou, Nandhakumar & Henfridsson, 2016; Tumbas et al., & vom Brocke, 2017).

A review of the entrepreneurship literature highlights aspects of the entrepreneurial context that may affect the work of designers in startups. To reduce the likelihood of failure and to succeed in building a new business, entrepreneurs must acquire financial resources (Van de Ven, Polley, & Garud, 1999) to fund software design, deploy marketing campaigns, and hire designers. The designers in a startup must also build a software to help support the digital business design. Startups are expected to operate under high uncertainty and, as a result, time to market is important (e.g. Huang, Rand, & Rust, 2016; Deeds, Decarolis, & Coombs, 1997). Designing software quickly provides startups with early cash flows, external visibility and legitimacy, early market share, and increases the

[1] I define design of the software as the entire process from analysis, design, coding, testing, maintenance and revision (Apte et al., 1990). Throughout design, interaction among the designers are focused on translating user needs into the design, converting the design into code and ensuring that different modules of the code work together without error (Sawyer, Guinan, & Cooprider, 2010).

likelihood of survival (Schoonhoven, Eisenhardt, & Lyman, 1990). This leads to my first research question: *How does the entrepreneurial context in which the startup operates affect the software design process and product in the early years?*

By entrepreneurial context, I am referring to the characteristics and design contexts of the software, particularly those occurring in startups. Context effects can be broadly defined as "situational opportunities and constraints that affect the occurrence and meaning of organizational behavior as well as functional relationships between variables" (Johns, 2006, p. 386). In this study, the focus is on the early years of the firm, which refers to the stage of the startup beyond the initial inception of ideas. At this stage, risks are taken in terms of investments, and the focus is on sales (Adizies, 1979). Adizies (1979) suggests that the early stage of a startup is an experimental phase in its life. Not only do designers in the startups face high financial pressures (Dahl, Nielsen, & Mojtabai, 2010), they also face high uncertainty resulting from the need to concurrently make sense of the unfolding business model with the novel software and capture emerging market opportunities (Hedman & Kalling, 2003; Chesbrough, 2012).

Additionally, entrepreneurship scholars have long underscored the relationship between entrepreneurs and investors as critical for the transfer of resources (Aldrich, Zimmer, & Jones, 1986) and for advice (e.g. Sapienza, Korsgaard, Academy, & Jun, 2007; Shane & Cable, 2002). Researchers have likened investors to large block stockholders in leveraged-buyout firms (Jain & Kini, 1995). The entrepreneur-designer relationship is a conduit though which financial and social resources critical to the survival of the startup flow (Shepherd & Zacharakis, 2001), yet their influence on software design is understudied. There are also entrepreneurs who choose not to seek investment because of the implications for product design so the entrepreneur's attention will not be directed by the investors towards growth and cashing out (Campbell, 2015). The entrepreneurs

interviewed describe the tension between building their brand and product versus taking actions to meet target revenues (Campbell, 2015). While software startups are proliferating, the influence of investors on software design is unclear.

The above discussion suggests that recognition of the early stage entrepreneurial contexts in which design occurs warrants further theorization. The entrepreneurial contexts challenge assumptions in existing software design literature: the existing software design research focuses on the design practices to help designers in large software design teams streamline their processes to meet user requirements (e.g. Chakraborty, Sarker, & Sarker, 2010; Maruping, Venkatesh, & Agarwal, 2009; Maruping, Zhang, & Venkatesh, 2009). Entrepreneurs in software startups often alter their business models based on opportunities (e.g. Kranz, Hanelt, & Kolbe, 2016; Ojala, 2015), which has implications for software design. At the same time, they operate in resource-constrained environments in the early years, so there is a tension between resource constraints and dedicating resources to designing new features that enable the capturing of emerging business opportunities. The context thus makes salient different sources of attention drivers that direct attention and resources of the designers in the design process.

1.1 Toward a Model of Software and Business Design in the Entrepreneurial Context

This dissertation is an account of how designers in software startups work in the early years of software design. The designers take on dual roles as entrepreneurs and designers of software. Given the exploratory nature of the research question, I draw on an eighteen-month, participant-observation case study of a three-year-old, digital health startup, HealthCom. The startup was designing a software to help nurses and patients communicate before and after patient treatments. HealthCom allowed me to study in real-

time how designers at a startup made design decisions based on environmental factors and stimuli.

I illustrate the presence of financial and software quality-focused attention drivers that directed the attention of designers. Attention to financial-focused design alternatives was driven by players, such as clients and investors of the firm. Clients refer to the healthcare facilities that licensed the software from HealthCom. Client intermediaries refer to administrators who participated in the purchase of the software at the client sites with whom the HealthCom designers maintain contact throughout the licensing agreement. The intermediaries dictated the success and renewal of contracts with HealthCom, and their requirements were therefore often prioritized by designers to ensure the financial performance of the firm. Investors, on the other hand, set expectations for firm performance for fundraising milestones. Attention to quality-focused design alternatives was driven by user requirements and designers' knowledge of software quality best practices.

Despite the tension between attention allocation to financial and quality-focused design alternatives, designers often moved from designing to accommodate existing user requirements towards designing to capture emerging business opportunities. In the case of HealthCom, the financial pressures led to the reframing of software was an enabler of funding, which meant designing software quickly and with minimal resources. These characteristics were at odds with designs for software quality. Designers had to balance the tensions between design decisions that could inhibit the startup's growth, compromise quality or over-optimize quality. The pressure for designs that contributed to funding led to designers leveraging design actions, such as simplifying of requirements, coupling of designs, and postponing of investment in design. I call the design product as a digital exoskeleton. The digital exoskeleton encompasses the following characteristics: (1)

disposability, (2) speed, (3) limited adaptability, (4) spartan functions, and (5) vulnerability. The work presented here is an attempt to outline a design process and product that captures the spirit and essence of the conflicts designers experience in the early stage entrepreneurial context.

The analyses presented do not serve to undermine existing theory, but rather offer an alternative view of software design that complements existing theory and research, helping to bridge gaps in the existing literature. The discussion on attention allocation in design is based on the principles of the attention-based view (ABV) theory. By looking at how the firm's environmental and organizational factors provide the general stimuli that direct the attention and actions of the designers, we can begin to understand when, why, and how designers decide what to design. Furthermore, an attention-based framework lends insight into how designers would respond under varying conditions. The rules of the game, the players, their structural positions, and resources within the firm all interact to affect attention allocation (Ocasio, 1997). What became evident in the analyses is that the attention of designers was influenced largely by investors and clients. The designers were subjected to two different sets of rules in software design and business design that resulted in design alternatives that frequently competed for the limited attention of the designers. The need to balance the tensions resulted in design actions that departed from typical software design approaches of Agile and Waterfall.

Applying ABV to design provides insights on the micro foundations of design in an entrepreneurial context and offers a lens for examining the impact of externally-oriented attention drivers on design actions and process. The model also explicates the impact of the early stage entrepreneurial context in intensifying financial pressures, uncertainty, and shaped design decisions, given the limited attention of designers.

1.2 DISSERTATION GUIDE

In chapter 2, I provide background on the IS software design research to highlight the gap in existing software design research. I also review the entrepreneurship literature to understand the contextual factors that might drive differences in software design in the entrepreneurial context. I illustrate how consideration of the ABV and entrepreneurship literature reveal gaps in the discussion on software design.

Chapter 3 explains the methodological foundation upon which this study was built. Specifically, I discuss the methodology, provide background on the research site, and describe procedures employed to collect and analyze data for this dissertation.

Chapter 4 illustrates the attention drivers directing the attention of designers and the tensions resulting from the financial and quality-focused attention drivers.

Chapter 5 further describes how the tension between the attention drivers resulted in design actions of reframing and simplifying requirements, coupling in design, and postponing of design effort. I conclude the chapter with an examination of how these actions culminated in the design product and a summary of what happened to the firm post-study.

Finally, in chapter 6, I explore the implications of the findings for theory of software design in entrepreneurial contexts and expand on the digital exoskeleton construct. I suggest that these findings can help researchers reconsider the impact of external players and rules of the game on software design and question the changing relationship between software quality and firm performance. I conclude with implications for future research on software design in the entrepreneurial context.

Chapter 2: Literature Review

2.1 Overview of Chapter

In this chapter, I begin with an understanding of the entrepreneurial context. I review existing IS software design literature for the key types of design processes (plan based and flexible approaches) and the benefits and challenges of the respective methods in design, particularly in the early stage entrepreneurial context. To better understand the impact of the entrepreneurial context, I leverage the attention-based view (ABV) as an organizing framework to understand how the different components of the entrepreneurial context (e.g. players, rules of the game, and resources) affect the attention of designers in the design process. Adoption of the ABV sheds light on how attention of the designers may influence their design actions.

2.2 Environmental Context in which Design Occurs

As mentioned in the introduction, this dissertation focuses on design that occurs in an early stage entrepreneurial context. In IS research, context has been referred to as the characteristics and use contexts of the technology artifact (Orlikowski & Iacono, 2001). By entrepreneurial context, I refer to the characteristics and design contexts of the software designed in startups, such as environmental and organizational factors experienced by designers in the startup. Context effects have been broadly defined as "situational opportunities and constraints that affect the occurrence and meaning of organizational behavior as well as functional relationships between variables" (Johns, 2006, p.386).

The study of context is important because it influences what individuals and groups see, decisions they make and the outcome (Autio, Kenney, Mustar, Siegel, & Wright, 2014). Yet there is limited agreement on what constitutes entrepreneurial contextual factors (Levie, Autio, Acs, & Hart, 2014; Zahra, Wright, & Abdelgawad, 2014). Scholars have

begun to isolate the dimensions of contexts that influence the actions of entrepreneurs: these factors include institutional, temporal, industry, market, spatial, social/organizational, ownership, and governance aspects (Levie et al., 2014; Zahra et al., 2014). For instance, technology innovation management scholars show that examining the industry life cycle can inform the difference in design focus of the firm: entrepreneurial activity in the early stages of an industry life cycle drive attention towards design of features and alternative designs (P. Anderson & Tushman, 1990; Utterback & Abernathy, 2003). The technological aspects of the context, which have been defined as the architectural attributes of the technology around which the entrepreneur acts (Thomas & Autio, 2012), also influence the actions of the designers and the stakeholders (thereafter called players) (Autio et al., 2014). These contexts change over time and shape entrepreneurs' actions (Wright et al., 2013). By considering the context, it focuses the attention beyond the individuals to examine how they direct attention to the various opportunities and threats (Alvarez & Barney, 2013; Autio et al., 2014) to understand the effects of the individual, situation, or serendipity.

While the relevance of contextual factors on design also applies in large firms (Zahra et al., 2014), the focus of this dissertation is in the entrepreneurial context, as much of the software design work is in the larger firms (e.g. Chakraborty, Sarker, & Sarker, 2010a; Lyytinen & Robey, 1999; Persson, Mathiassen, & Aaen, 2012).

Even though researchers have discussed the role of context in influencing their findings, significant challenges remain in integrating context in theory (e.g., Bamberger, 2008; Johns, 2006). For instance, context can serve as a main effect, have cross-level effects, interact with other variables, change the situational strength of factors (Johns, 2006), or abet or constrain human agency (Mischel, 1968). Context can also be understood as a configuration or bundle of stimuli that may be hard to tease apart but still yield "a more

interpretable and theoretically interesting pattern than any of the factors would show in isolation" (Rousseau & Fried, 2001, p.4). Johns (2006) asserted that the challenge with studying contextual factors is the lack of emphasis of the interactions between these factors. These contextual factors are often studied in isolation, with other aspects of the context excluded from the study, even though they could be interdependent.

2.2.1 Stage of Startup of Interest

Scholars and practitioners also recognize that there are different phases in the process of entrepreneurial design of new products and software that varies with the organizational lifecycle (e.g. Gersick, 1994; Vohora, Wright, & Lockett, 2004). In this study, the focus is on the early years of the firm, beyond the initial inception of ideas. At this stage, risks are taken in terms of investments and the focus is on sales (Adizes, 1979). Adizies (1979) suggest that in this stage of the firm, the structures and processes within the firm are coalescing and that this is an experimental phase which demands continuous commitment of the founder, who is the only management body at the time. Beyond this stage, the attention is not sales but on growth. Startups in nascent markets are in the exploratory stage, in which startups have to learn, make sense of, and adapt to changes in the market and firm, while needing to stay dynamic, flexible, innovative, and agile (e.g., Eisenhardt & Martin, 2000)

2.2.1.1 Salience of Firm Performance in the Early Years

Entrepreneurship scholars have emphasized the high failure rates of new firms (e.g., Freeman, Carroll & Hannan, 1983) and the negative repercussions of failure for the startup's employees (Agarwal & Audretsch, 2001; Decker, Haltiwanger, Jarmin & Miranda, 2014; Burton, Dahl, & Sorenson, 2017). The fear and stresses of failure (Dahl,

Nielsen, & Mojtabai, 2010) have resulted in increased attention to firm performance in the discussions and decisions.

Additionally, attention to firm performance is affected by the involvement of investors that can influence decisions in the firm. Collewaert and Sapienza (2016) suggest that while investors and entrepreneurs generally work together towards maximizing the value of the startup, they may disagree on how to do so. Disagreements ensure about focusing attention on product development, gaining technology superiority, market share, or additional financing, all aspects which are essential to a venture's progress (Fiet, 1995; Politis, 2008). Venture capitalists can also hinder venture growth if they offer the wrong strategic input or impose ill-advised constraints (Gomez-Mejia, Balkin, & Welbourne, 1990; Steier & Greenwood, 1995). Venture capitalists distinguish between different investment stages with financing provided for different purposes. But despite the proliferating venture-backed software firms, there is little recognition of the stages on the process of software design. Given the pressure for entrepreneurs to demonstrate value and generate sales in the early years, entrepreneurs may make decisions that can be myopic and detrimental for design in the long run (e.g., Das & Teng, 1998). Thus, the design decisions of startups in the early years warrant further study.

2.2.2 Changing Nature of Software Designed

Software designers in startups face pervasive uncertainty regarding market acceptance, the ability to mobilize resources, recognition of opportunities and whether their designs will actually work (Autio et al., 2014). The uncertainty they face is exacerbated by the changing of digital technologies, such as platform designs (Yoo, Henfridsson, & Lyytinen, 2010) or cloud computing, that shape the way software is designed (Lehmann & Rosenkranz, 2017). These technologies are viewed as a combination of "information,

computing, communication and connectivity technologies" (Bharadwaj et al., 2013). In the case of design processes in early stage startups, new digital technologies introduce new challenges in developing a business model, as the software may be designed prior to identification of a market, resulting in uncertain or difficult assess market potential (Shepherd, Mcmullen, & Ocasio, 2017; Sydow, Schreyögg, & Koch, 2009, von Briel et al., 2017). Designers in startups may continue to receive additional information that changes their attention allocation to design alternatives. (Antonopoulou et al., 2014). The market for the software, and the absence of a business model upfront (Chesbrough & Rosenbloom, 2002) results in uncertainty for entrepreneurs. Therefore, designers in early stage startups would not only have to manage the uncertainty of the design process as in large software design firms (Lyytinen, 2001; Tuomi, 2002; Hanseth & Lyytinen, 2010), but also concurrently make sense of the unfolding business model and capture emerging market opportunities (Hedman & Kalling, 2003; Chesbrough, 2012).

Instead of a relatively stable product and well-defined service boundaries (Davidsson, 2015; Nambisan, 2016), the design products underlying entrepreneurial opportunities are intentionally incomplete and can be expanded even after a product has been shipped (Garud, Jain, & Tuertscher, 2008; Yoo et al., 2010). For instance, in the case of a startup trying to find a market for its software, its software continues to evolve as it obtains more clients (e.g., Antonopoulou et al., 2016, 2014). Antonopoulou et al. (2016) illustrate the use of reusable designs that can be repurposed for other business contexts critical for firm performance in the early years of a startup. Overall, the changing digital technology (Nambisan, 2016) has improved the ease of capturing new business opportunities outside existing functionalities and contexts (e.g., Kranz et al., 2016; Ojala, 2015) but can still be unwieldy for entrepreneurial firms to manage in the early years.

When designers and users struggle to make sense of the value of the software, this often results in ambiguity in quality and requirements. The design process in a startup can "unfold in a nonlinear fashion" (Nambisan, 2016, p. 6). Unlike the traditional software design firm, the designers in startups undergo greater extents of experimentation and changes in design trajectories (Nambisan, 2016). The goal is thus not to design a scalable robust software in the early years but to identify, through existing design efforts and exploration of opportunities, a scalable and repeatable business model (Blank, 2013).

While the nature of the software and the environmental context can influence the software design process, IS scholars have yet to adequately consider the contingencies surrounding processes of designing software in entrepreneurial firms. Unlike large firms, these early stage startups operate under high uncertainty and time and financial pressures, as they concurrently design and search for product market fit.

Because of this uncertainty, designers have been observed to postpone investment in design. For instance, Tumbas et al. (2015) adopted a digital façade concept that describes how mid-stage entrepreneurs have leveraged digital technology to give the appearance of an established and mature firm. Behind the scenes, the designers improvise with manual, ad hoc processes to meet the needs of their clients. The adoption of a digital façade is an example of a minimal commitment in design while waiting for uncertainty to resolve. However, the digital façade refers to digital technology that supports the operations of the startup but is not the core product offering of the startup. The digital façade example begins to highlight the possibility that startups in the early years may adopt design actions at odds with the espoused software design practices.

2.3 Existing Methods of Software Design

While the design process encompasses analysis, design, coding, testing, maintenance, and revision (Apte et al., 1990), one of the key foci in the software design literature in IS is the analysis process, in which scholars examine the problems in requirements elicitation and changing requirements (Iansiti & MacCormack, 1997; Lee & Xia, 2005; Maruping et al., 2009; Banker & Slaughter, 2000), or coordination within designers (Faraj & Sproull, 2000; Maruping, Zhang, et al., 2009). For instance, Guinan et al. (1998) describe the problems with users' inability to specify requirements accurately (e.g., Boland, 1978) and identify problems with designers who were unable and unwilling to work with users (e.g., Lyytinen & Newman, 2015), resulting in a final product that failed to meet users' needs (Guinan et al., 1998). Guinan et al. (1998) also acknowledge the complexity of the requirements resulting from the elicitation/determination process, the importance of creating common goals, managing team progress, and failure and team culture. Guinan et al. (1998) further suggest that it is not the tools or software design methods but people, skills, and team characteristics that affect team performance. To that end, to better understand how to elicit user requirements and manage the changing requirements, software design practitioners have created two key types of software design methods to manage the costs of changing and/or inaccurate user requirements: plan-based and flexible design approaches.

2.3.1 Plan-Based Methods of Software Design

2.3.1.1 Practices in the Waterfall Method

In the plan-based approach, such as the Waterfall method, there are distinct phases of specification of requirements, design, implementation, and testing (Sommerville, 1996).

However, a designer can only move to the next phase after the completion of the previous phase (Pressman, 1997; Royce, 1970). In the specification of requirements phase, the designers elicit and capture the requirements of users in a product requirements document, which is verified before design occurs. Design results in the creation of the software architecture and coding of the design. The implementation and testing then occurs, as the designers systematically discover and resolve the defects (Royce, 1970). In modified versions of the Waterfall model, designers can return to the previous cycle if defects or problems have been identified downstream (Royce, 1970).

2.3.1.2 Benefits of the Waterfall Method

Planned based approaches are effective when software requirements are fully specifiable, predictable, and can be built through meticulous and extensive planning management (Dybå & Dingsøyr, 2008). Because of the emphasis on documentation through requirements documents and design documents and source code, it ensures continuity in design if team members were to leave midway during the design process or for other designers to understand the code. Plan-based methods are also more relevant for requirements that remain stable. Additionally, designers have to pay attention to strict budgetary or schedule constraints, or quality, for software that require high reliability, such as safety-critical systems (Maruping et al., 2009). Attention to these aspects of design thus leads to a design process that is largely plan-based, as the plan will outline the requirements, architectures, designs, budgets, and schedules with little deviation (Maruping et al., 2009).

2.3.1.3 Challenges with the Waterfall Method

Critics of the Waterfall approach point to the difficulty in eliciting accurate user requirements upfront (Lyytinen, 1987; Sommerville, 1996). The plan-driven, formal and tool-oriented Waterfall approach often leads to unnecessary redesign downstream after significant investment in design and implementation (Conboy, 2009). The long lead times can lead to redesign, redesign, retesting, and increased costs (Cooper & Woolgar, 1994). Risk is therefore high for developers, as it will not incorporate newly-discovered constraints, requirements, or problems in the meantime (Parnas & Clements, 1986). The risks suggest that adoption of the Waterfall method in the entrepreneurial setting with high uncertainty and evolving and emergent players will result in requirement documents and design plans that will quickly become obsolete.

2.3.2 Flexible Methods of Software Design

2.3.2.1 Practices in the Agile Method

Flexible design methods, such as the Agile method (e.g. Abrahamsson, Conboy, & Wang, 2009), recommend organizing design processes in rapid prototyping cycles. The Agile method shifts the attention away from practices common in plan-based design methods: it prioritizes the creation of working software over comprehensive documentation, user collaboration over contract negotiation, and responding to change over following a plan (Beck, 2004).

The Agile method also recommends design practices, such as paired programming, daily standup meetings, depending on the variant of Agile method (e.g. Scrum, XP). Agile methods recommend using prototypes as validation mechanisms to avoid waste in upfront design, particularly in complex projects (Cao, Ramesh, & Abdel-Hamid, 2010). Prototypes

are used for establishing a common basis for understanding and communicating design ideas between a designer and the players (Mathiassen, Seewaldt, & Stage, 1995), enabling the designer to respond quickly to changing requirements (J. Highsmith & Cockburn, 2001). Planning only happens in detail for the features and requirements to be implemented in a specific cycle. These practices enable flexibility to incorporate changes in a late stage of the project with less consequences and improve the ability to demonstrate business value more quickly (Dagnino, Smiley, Srikanth & Ant, 2004) and improve the value of the software to users (Beck, 2004). The design process is thus adaptive and iterative (Lee & Xia, 2010).

2.3.2.2. Benefits of the Agile Method

In principle, flexible approaches are good for complex systems and projects with dynamic, non-deterministic characteristics that render accuracy in estimates useless, as feedback is frequently sought from users that will then be used to guide further design (Maruping et al., 2009). While the Agile method is also said to be developed for small, co-located teams (Boehm & Turner, 2005; J. Highsmith & Cockburn, 2001) due to the extent of daily coordination, use of Agile methods has been adapted for larger team sizes (Cao, Mohan, Xu, & Ramesh, 2009; Brian Fitzgerald, Hartnett, & Conboy, 2006; Maruping, Venkatesh, et al., 2009) as a result of desire for agility in design. Adoption of Agile practices has increased due to challenges resulting from a changing environment, ambiguous user requirements, and time constraints (Boehm, 2002; Maurer & Melnik, 2006; Abrahamsson & Still, 2007).

2.3.2.3 Challenges with the Agile Method

Although the Agile method seeks to deal with uncertainty to generate innovations at a high rate, it requires the adoption of many routines (Berente & Lyytinen, 2007; Fitzgerald, 1997) that an early stage entrepreneurial firm can find too resource-intensive to adopt. Additionally, Conboy (2009) asserted that designers can adopt Agile method practices but still not achieve agility in the software design process. For instance, he saw that the use of standup meetings, while beneficial, had a varying impact on agility. Other scholars have also found that adoption of Agile practices used is not necessarily correlated with the achievement of agility because the context of use and implementation process matters (Cao et al., 2009; Conboy, 2009; Vidgen & Wang, 2009).

The Agile method is also limited in its benefits as it has been normative, in that it has a list of prescribed practices that have little clarity with regards to its contribution to agility and theoretical grounding. In fact, scholars have found that designers often do not adopt Agile practices holistically, not because of ignorance but for pragmatic reasons (e.g., Fitzgerald, 1997; Conboy, 2009). In fact, scholars have suggested that a software design method may be interpreted, not as a set of rules or rigorous prescription, but as an ideal that is not expected to be followed literally (Conboy, 2009; Iivari & Maansaari, 1998).

In the entrepreneurial context, the use of Agile methods can also be limited, due to the novel nature of the software. Scholars have shown that the value of user interactions in Agile is limited when users have insufficient knowledge of the requirements due to the complexity of the designs or lack of user participation (Fitzgerald et al., 2006). Users and/or client intermediaries need to be available frequently (Highsmith, 2002), but in the startup in the early years, the startup is still trying to identify its product-market. As such, it does not yet have a fixed user group but one that may change frequently over time. Additionally,

the Agile method also often assumes an "ideal" user representative, who can answer all developer questions correctly and is empowered to make binding decisions correctly (Paetsch et al., 2003). Communication may be ineffective or resource intensive when dealing with many players and vast amounts of information (Fitzgerald et al., 2006).

Additionally, the assumption is that designers in the Agile method will incorporate changing requirements in the design (Beck, 2001). However, scholars have found that updated requirements may only be incorporated if designers have time (Lee & Xia, 2010). Requirements may change frequently, which leads to a high overload cost for designers to get frequent testing at the end of each sprint (Beck, 1999). Environmental uncertainty for the designers is higher compared to that in the plan-based methods, as user requirements and specifications can change over the course of a project (Maruping, Zhang, et al., 2009; Nidumolu & Subramani, 2003).

The use of Agile methods also has negative repercussions for the software product, as a result of inadequate architecture planning, overemphasis on early results (E. Anderson, Lim, & Joglekar, 2017), and low levels of test coverage (Highsmith & Cockburn, 2001; Boehm, 2002). Additionally, the use of Agile over time can lead to a lack of architectural scalability, as the absence of a design phase may lead to the overlooking of design problems upfront with consequences downstream (Erickson, Lyytinen, & Siau, 2005). The use of flexible methods, such as Agile, have been posited to increase the intensity of technical debt beyond that discussed in traditional software design (Elbanna & Sarker, 2016). Because of the need to reduce design time and focus on the delivery of functional requirements, designers have postponed the repayment of technical debt.

Overtime, use of the Agile method results in trouble meeting schedules and lowers software quality, as technical debt accumulates, and software grows in complexity and becomes more difficult to maintain. Additionally, the focus on designs associated with

business also means reduced attention to stability and reliability in the software. The problems with use of Agile method highlighted by scholars thus contradicts Tumbas et al. (2017), who suggest that startups have little technical debt in the early years. Collectively, the research suggests that flexible methods, such as Agile, can result in software with different risk and success factors from traditional software design (Elbanna & Sarker, 2016) that may present opportunities and constraints for startups in the early years trying to concurrently design a software and a business.

2.3.2.4 Practices in Lean Startup, Another Flexible Method of Software Design

A related flexible software design approach espoused for startups is the Lean startup (Ries, 2011), particularly for startups navigating nascent markets that face high levels of uncertainty and ambiguity (Santos & Eisenhardt, 2009). The Lean startup is grounded in Lean thinking. Lean thinking focuses on cost reduction and quality and has shifted to focus on increased value (Conboy, 2009). The value refers to the value experienced by users through the product offering (Ries, 2011), instead of value from the designers' perspective.

2.3.2.5 Benefits of Lean Startup

The impact of the entrepreneurial context (Tumbas et al., 2017, 2015) has changed how software products are designed. Practitioners have recommended prototypes, such as the minimum viable product (MVP), for designers in startups. According to Eisenmann, Ries and Dillard (2012), p. 1), "an entrepreneur translates her vision into falsifiable business model hypotheses, and then tests those hypotheses using a series of MVPs. Each MVP represents the smallest set of activities needed to disapprove a hypothesis." An effective MVP design approach requires rigorous discipline approach to generate and test

hypotheses. Hypotheses are tested in short prototyping cycles. The objective of the MVP is to maximize the amount of learning and uncertainty reduction with the minimum resources expended. The notion of prototypes common in the flexible appeals to entrepreneurs operating in a resource-constrained environment (e.g., Ries, 2011). It involves relatively little upfront investment in the design process but focuses on iterations to allow entrepreneurs the opportunity to validate their business models efficiently. Note that the goal of the design effort advocated in the Lean Startup is not to build robust software but to validate and identify a scalable business model (Blank, 2003).

2.3.2.6 Challenges with Lean Startup

The focus of the Lean Startup is thus on the design of single features and conducting experiments, such as split testing, to verify the desirability of the feature (Ries, 2011). The interdependencies between the features and cohesiveness of the features as a whole are thus secondary (Sharkey, 2013). Additionally, because of the recommended frequency of experiments and feedback evaluation (Ries 2011), the process is resource intensive, which bears similarities to the problems with the Agile method but intensified.

2.3.3 Summary of design Methods

In general, these software design methods do not provide guidance to designers on how to approach "dynamic, chaotic, multi path and expansive" (Yoo et al., 2010, p.7) design of digital technology often designed in startups. A literature review by Park, Boland and Yoo (2011) indicate that only 12 percent of existing design process research in the top IS journals have focused on the syntheses of new ideas, forms, and functions in the design process. The majority of existing design process research examines the analysis and evaluation of design requirements from available user groups but designers in software

startups tend to move towards synthesis path creation (Henfridsson, Yoo, & Svahn, 2009) or recombination (Hylving, Henfridsson, & Selander, 2012). Designers in the early stage entrepreneurial context create novel software with new, emergent value propositions and ambiguous quality that are undetermined upfront (Antonopoulou et al., 2014). The Lean Startup and other flexible approaches appear to partially alleviate the uncertainty but may be resource intensive (Fitzgerald, 1997; Berente & Lyytinen, 2007). These flexible design approaches also highlight a disregard for architecture and design principles (Sharkey, 2013).

2.4 Examining These Challenges Through ABV

In this dissertation, I adopt the attention-based view (ABV) as an organizing framework. Adopting an ABV allows researchers to understand how designers allocate attention given to the attention structures, environment, and communication and procedural channels that exist within and outside of the firm. The plan-based and flexible approaches highlighted in the previous section have highlighted different design process practices for uncovering the user requirements accurately and efficiently. In the remaining parts of this chapter, I provide a summary of the ABV before addressing the value and limitations of adopting ABV as an organizational framework.

2.4.1 Attention-Based View

In this section, I provide an overview of ABV, a framework that emerged through the analysis of my data. The rationale for ABV is that decision-makers can attend to only a limited number of issues and answers due to time and cognitive limits (March & Simon, 1958; Simon, 1947). The ABV describes how attention is funneled through the attention structures and communication and procedural channels to focus on the concentration of attention and resources of decision makers on a limited set of issues and tasks, thus

facilitating the speed and accuracy of decision-makers' response (Ocasio, 1997). Attention is defined as the decision-makers' noticing, encoding, interpreting, and focusing of time and effort on issues and answers (i.e., the repertoire of categories for making sense of problems, opportunities, and threats in the environments, and an available repertoire of action alternatives that could be used to address the issues) (Ocasio, 1997).

The attention structures comprise four components: players, rules of the game, structural positions, and resources (Ocasio, 1997). Players are structurally autonomous social actors or groups of actors, who, through their social influence, power, and control, influence and regulate the decision and activities of other decision-makers (Ocasio, 1997). Rules of the game are formal and informal principles of action, interaction, and interpretation that guide and constrain decision-makers in accomplishing the firm's tasks, and in obtaining social status, credits, and rewards. They embody the organizational identity and purpose (Barnard, 1938; Dutton & Dukerich, 1991; Selznick, 1957). Structural positions are the roles and social identifications that specify (a) the functions and orientations of decision-makers, and (b) their relationships with other structural positions internal and external to the firm. They arise from the division of labor both within and between organizations. Lastly, resources are tangible and intangible assets that allow the firm to perform its activities and to produce its goods and services (Wernerfelt, 1984). Together, these four components shape the legitimacy of available issues and answers and help to funnel the attention of decision makers to the selected set of issues and answers (Fiol & O'Connor, 2003; Simon, 1947; Starbuck & Milliken, 1988).

2.4.1.1 Attention Structures: Players

Players are "individuals [who] ultimately do the attending" and have considerable influence over the attention regulation within the firm (Ocasio, 1997, p. 189). They regulate

the firm's attention through the specific skills, beliefs, and values they bring to the firm (March & Olsen, 1975). Ocasio asserts that any factor that influences "the agenda for the meeting" or "the formal structure of the committee" is likely to be very important (1997, p. 195). Research has included those in top management teams (Adams, Licht, & Sagiv, 2011; Cho & Hambrick, 2006; Van Doorn, Jansen, Van Den Bosch, & Volberda, 2013) and board members (Tuggle, Schnatterly, & Johnson, 2010).

In the software design literature, examining the players in the process is important because players bring different goals and perspectives into the design process, as evidenced by the studies in the software implementation process (e.g. Thanasankit, 2002). However, existing software design research views software design as grounded in software engineering principles, thus is less people centric (Nerur, Mahapatra, & Mangalaraj, 2005).

2.4.1.2 Structural Positions

Structural positions refer to "the roles and social identifications that specify (a) the functions and orientations of decision-makers, and (b) their interrelationships with other structural positions internal and external to the firm" (Ocasio, 1997, p. 197). Structural positions create heterogeneity in the interests and identities of decision-makers and thus result in "differentiated attention to various aspects of the organization's environment" (Ocasio, 1997, p. 198). Depending on the structural positions and interests associated with those positions, decision-makers will weigh some aspects more heavily than others. Scholars further propose that differences in roles and responsibilities create differences in motives across individuals in different structural positions, which shape attention allocation (Barreto & Patient, 2013; McMullen, Shepherd & Patzelt, 2009). These studies suggest that the structural positions of the designers and the roles they play can affect attention allocation in the design process.

2.4.1.3 Rules of the Game

The rules of the game are "the formal and informal principles of action, interaction, and interpretation that guide and constrain decision makers in accomplishing the firm's tasks and in obtaining social status, credits, and rewards in the process" (Ocasio, 1997, p. 196). The rules include the rules competitors subscribe to, the basis of competition, and other socially constructed rules that are espoused by decision makers or other players. In the organization sciences research, Cho and Hambrick (2006) found that a change in the "rules of the game" (Ocasio, 1997) as a result of deregulation in the airline industry led decision makers to shift from an internal orientation towards an external market orientation, which led to the noticing of new opportunities for the creation of value. Just as subunits within a firm have different goals (e.g., production department focuses on production goals and marketing department on marketing goals), it can be imagined that the decision makers or subunits within the firm may be sensitized to different attention structures. Thus, further insight on the influence of the rules of the game on attention allocation may shed insight into how attention and resources become allocated in the design process.

2.4.1.4 Resources and Firm Performance

Resources are assets used in the construction of the firm's action (Ocasio, 1997) and can refer to human, physical technology, or financial capital (Ocasio, 1997). Firm performance affects resources available to the firm and the alternatives that can be considered by the decision-makers (March & Shapira, 1992; Ocasio, 1997). These include competencies and inimitable assets (Rumelt, 1984; Selznick, 1957). With more resources, attention can be diverted away from the prioritized issues and distributed among multiple issues (Chen & Miller, 2007; Sapienza, De Clercq, & Sandberg, 2005). It can be imagined that this becomes particularly salient for startups that are resource-constrained.

Startups without slack resources will first allocate attention to actions that allow the achievement of urgent issues (George, 2005), such as improvement in financial performance. Work that does not contribute to the financial targets is abandoned, though less frequently when the firm has resources (Shimizu, 2007). Abundant slack resources create a buffer for decision-makers to allocate attention to other non-urgent issues. Slack promotes experimentation with new strategies, shapes the consideration of alternatives, and influences the repertoire of answers of the decision-makers (Ocasio, 1997). Slack resources further influence the attention of decision-makers by including (or excluding) different alternatives, as the availability of resources provides the organization with the possibility to perform a wide variety of tasks (March & Shapira, 1992), ideas, and projects that otherwise would not be approved.

2.4.1.5 Procedural and Communication Channels

Procedural and communication channels include in-person or virtual meetings, telephone conversations, and email exchanges, etc. They are the conduits that guide the attention allocation of decision-makers within the firm (Ocasio, 1997). The interactions between decision-makers through these channels allow discussions about specific activity or situation knowledge, alternative answers, interests, and identities. These interactions shape which issues and answers become more salient and direct decision-makers to attend to different components of their environment (Sutton & Hargadon, 1996).

2.4.2 Attention Dictates Action

Attention allocation amongst designers within the firm is shaped by attention drivers and contextual factors, such as firm performance or time constraints. The attention allocated then affects action (Ocasio, 1997). For instance, in the software design research, Abdel-Hamid et al.(1989) studied the association between goals and project actions. The

findings suggest that, with attention to a given project goal, team managers make planning and resource allocations to meet such a goal. Designers who have to meet cost and schedule constraints allocate different attention and resources to designs than those who have to deliver on quality and schedule. Lee and Xia (2010) find that when faced with time constraints, designers will continue to pay attention and accommodate high impact, business-disruptive requirement changes, despite the repercussions for software quality. Paying attention to meeting changing user requirements may lead to actions that increase the expense of budgets and schedules (Maruping et al., 2009). Overall, these studies show that attention to certain aspects of design will lead to allocation of resources towards those aspects.

Attention allocation can affect design actions. If uncertainty is low, designers may adopt a plan-based design approach to quickly complete the design. With time constraints, a flexible approach will not be faster, as it can result in redesign (Maruping et al., 2009). The pressures of user expectations, time, and resource constraints have led to limitation of scope (e.g., Siddiqi & Shekaran, 1996) and delivery of the most essential functionalities as early as possible (Port & Bui, 2009) that may depart from product market fit (Harris et al., 2009). Collectively, these studies show that what the designer pays attention to in the design process can affect the actions undertaken, though there is little understanding of the specific design actions undertaken when the designers' attention may be driven by different attention drivers, particularly under entrepreneurial contexts with high uncertainty and high financial and time pressures.

2.5 UNDERSTANDING DESIGN IN THE ENTREPRENEURIAL CONTEXT THROUGH ABV

2.5.1 What ABV Informs us about the Entrepreneurial Context

To begin to answer my research question of "*How does the entrepreneurial context in which the startup operates affect the software design process and product in the early years?"*, my review highlights that IS research has focused on software design processes and practices, such as Waterfall (e.g. Sommerville, 1996), Agile (Abrahamsson, Conboy, & Wang, 2009; Conboy, 2009; Highsmith, 2002), and other software design processes. The goal of these practices includes control of software design process efficiency (Zmud, 1980), budget and schedule (e.g. Cao, Ramesh, & Abdel-Hamid, 2010; G. Lee & Xia, 2010), whilst meeting users' requirements. The studies have also shown that attention to factors, such as budget control or time pressures (e.g., Austin, 2001; Abdel-Hamid et al., 1989), shape the design process. ABV highlights how players and rules of the game in the entrepreneurial context have the potential to change the dynamics in which design discussions occur.

In the software design literature, a software that does not meet user needs is a failure, according to software quality definitions (Guinan et al., 1998). The software design rules (e.g., design and programming guidelines, security, and assurance guarantees) aim to produce a coherent design product (Harris, Collins, & Hevner, 2009) Design practices, such as increased designer-user communication, have been adopted to increase user satisfaction, quality of software, and greater efficiency for the designers. In the Waterfall approach, the rationale is that accurate elicitation of requirements in the early stages will reduce any redesign in the latter stages (Royce, 1970). Conversely, the Agile method recommends organizing its design processes in rapid prototyping cycles to reduce the need

for redesign(e.g., Abrahamsson, Warsta, Siponen, & Ronkainen, 2003; Dybå & Dingsøyr, 2008).

Examining ABV with respect to software design illustrates how designers shape the procedural and communications channels within the firm based on the rules of the game they are playing. For example, to capture more accurate user requirements, the procedures have changed from upfront planning to iterative design in Agile methods to allow for more frequent user feedback (Conboy, 2009). The procedures within the Agile methodology and requirements engineering literature also focus on increased user participation in the design process. The user is the focus of attention in the design process. However, as mentioned in the earlier sections, with the emerging digital technologies (Nambisan, 2016; Tiwana et al., 2010) and increasingly turbulent software design environments (Vidgen & Wang, 2009), typified by unpredictable markets, changing user requirements, and time pressures (Baskerville et al., 2001), more research is warranted to see how designers, such as those in the early stage startups, allocate their attention and design efforts. Additionally, when a firm, such as a startup, is resource-constrained, the designers' attention tends to be preoccupied with short-term performance issues, rather than with long-term, innovative projects (George, 2005).

Furthermore, the existing research focuses primarily on users and designers. Many scholars have argued for the importance of designer-user interactions and mutual understanding between designers and users (Kujala & Väänänen-Vainio-Mattila, 2008; G. M. Marakas & Elam, 1998). User-designer interactions can improve the technology's quality by matching the task needs of the user with the technology capabilities (i.e., task-technology fit) (He & King, 2008). As such, the software design practices often focus on communication channels and other procedures that allow the designer to elicit accurate information about user requirements to improve software success (Gallivan & Keil, 2003;

George M Marakas, Yi, Johnson, & Introduction, 1998; Schonberger, 2010). For example, XP (a form of Agile method) insists on having an on-site customer (Beck, 2003) whose responsibilities include understanding and representing the needs of multiple customer segments and specifying and clarifying the features that need to be implemented. Some IS scholars have suggested that users, user representatives, and their intermediaries are less influential in the case of enterprise packaged software or novel software (e.g. Sawyer, 2001; Stacey & Nandhakumar, 2006), such as the one designed in this dissertation.

The software design literature does not focus as much on client intermediaries (e.g., administrative staff from client sites), though the literature questions the extent to which the intermediary is acting as reliable representatives of the users (Iivari & Iivari, 2006; Majchrzak & Beath, 2001). For instance, systems analysts and designers are the voices of the customers, and they communicate their learning to the other designers in the form of storyboarding or use cases to inform design. The client intermediaries, such as those who are responsible for purchasing the software, are unavoidably intertwined with the politics of resource allocation and legitimacy of decision-making within their own firms (Keil & Carmel, 1995). For instance, client intermediaries who sought to protect their own careers, advocated inefficient designs that seemed to reflect their own self-interests rather than for the benefit of other players (Chatterjee, Sarker, & Fuller, 2009). Chatterjee, et al. (2009) showed that a powerful union negotiated several conditions, including attractive retirement benefits for staff who would retire following the software implementation. As such, the input of the existing staff is not incorporated or deemed relevant for the design of software. Despite this recognition, the software design literature does not investigate the challenges of designing software with client intermediaries as players influencing the design process.

2.5.2 Gaps in our Understanding of the Entrepreneurial Context and Attention Allocation

However, in the ABV the interactions between the components of the attention structures, channels, and environment are black boxed and thus presents gaps in the understanding of the relationships between attention allocation, design actions and product in the early stage entrepeneurial context. Currently, the assumption is that the ABV integrates the attention of individuals within the firm (Ocasio, 1997). The interactions can change in consideration to the entrepreneurial context, due to the presence of different players and rules of the game.

Using the ABV highlights other players, such as designers, clients and, indirectly, investors who have influence over attention allocation through their control of the resources available to the firm. For instance, a review of the entrepreneurship literature suggests that elements of the entrepreneurial context, such as the involvement of investors (e.g., Sapienza, Korsgaard, Academy, & Jun, 2007; Shane & Cable, 2002), may affect the work of designers in startups. The financial pressures and uncertainty of the entrepreneurial context may shape attention allocation and action: given the smaller firm size and dual roles of designers as entrepreneurs.

2.5.1.1 Designers

Information systems scholars have examined how the designers and their perspectives affect the design process. For instance, designers have been found to be unwilling to work with user-provided requirements because they think they know what is best for the user (Lyytinen & Newman, 2015). Designers may allocate attention differently depending on their personal goals. Austin (2001) further suggests that designers who pay attention to professional or financial gains fear the career impact of missed deadlines more

than quality tradeoffs and may take shortcuts. These shortcuts reflect designers' "tendencies to hope for the best, to leave potential sources of difficulty unexplored, and to interpret requirements conveniently when faced with time pressures" (Austin, 2001, p. 195). Designers take shortcuts without fearing personal consequences because it is difficult for others to identify the root of defects or performance issues (e.g. DeMarco, 1995; Iansiti & Gill, 1990). Designers are also found to allocate attention to different aspects of design, depending on their roles. Designers responsible for writing and testing the code "often have different and even divergent goals that are difficult to synthesize" (Sawyer, 2001a, p. 160).

Less is said about the designer who has to take on multiple roles, such as design and business design (Onyemah, Pesquera, & Ali, 2013). Whereas existing software design research occurs in large organizations where buffers may exist between business design and software design, designers in the entrepreneurial context may have to take on multiple roles. Entrepreneurs who take on multiple roles within the firm have to navigate multiple identities (Mathias & Williams, 2014), all of which can affect attention allocation.

2.5.1.2 Other Players

Several scholars have begun to delve into the social processes and actors involved in the design process (e.g. Chakraborty, Sarker, & Sarker, 2010). Players are now more distributed across space, stakeholder groups, and time (Hansen & Lyytinen, 2010). Research alludes to a presence of players in the market who can affect the design process, for example, by influencing the speed at which designs have to be (e.g. Huang et al., 2016). Not recognizing these players would result in "overlooking the most critical problems with [software design], those of different expectations and reactions of different [players]" (Chatterjee et al., 2009, p. 790). Examining software design using the attention-based view recognizes the influence of players in attention allocation of the decision makers within the

firm. Acknowledgement of these players shapes dynamics among the designers that is needed for improved software design performance (Carmel & Sawyer, 1998).

The organization science and entrepreneurship literature suggest that players external to the firm may be involved in attention allocation within the firm. In the organization science literature, researchers have examined how players and relationships at the industry level (e.g., Hoffman & Ocasio, 2001), and at the firm level (e.g., Kaplan, 2008; Tuggle et al., 2010) can affect attention allocation. Similarly, the entrepreneurship researchers have emphasized the role of investors in the running of the startup. Given the substantial equity stake that venture capitalists typically take in a firm, the relationship between venture capitalists and the entrepreneurs that they fund has been compared to that of large block stockholders in leveraged-buyout firms (Jain & Kini, 1995). Venture capitalists observe the firm to track its business potential and monitor the entrepreneurs' actions (Cable & Shane, 1997). Venture capitalists also introduce new constraints for the firm and the design process. Venture capitalists face time pressures because their constituents expect returns on their investments. Time is an integral component of financial returns, as the venture capitalists' performance is measured by total returns over a fixed time. Thus, venture capitalists experience pressure to move products to market and grow ventures to profitable stages (Cable & Shane, 1997). Given the influence of the investors on the running of the business, their involvement cannot be ignored, yet it has not been acknowledged in the software design literature, despite the growing phenomenon of software entrepreneurship.

Overall, the research cumulatively highlights the need to examine the broader set of players and their influence in the design process. Guinan et al. (1998) describes the need for activities to help designers monitor and restrict the external influences in the design

process. However, Guinan et al. (1998) only focuses on the requirements elicitation phase, not what happens beyond the requirements elicitation process.

2.5.1.3 Rules of the Game

As alluded to previously, designers in the startup may adopt distinct design actions or create different design products than those in the established software design firms (e.g., Nambisan 2016). To better understand the sources of environmental influence experienced by startups, a review of the practitioner literature suggests that the designers in startups have to play an additional but different "game." Other than the extant software quality and project management rules, startups have to meet funding milestones and growth metrics that are different than in software design, as highlighted in Table 2-1. Table 2-1 summarizes the rules of the game for designers pertaining to software design and digital business design. The first two rows highlight software quality and project management outcome measures that have been discussed in IS research. The startup growth and funding milestones highlight the financial rules by which software startups are evaluated, particularly in the early stages. Startup performance is determined based on these financial metrics, which do not explicitly consider quality but have implications for resource and attention allocation within the startup. These additional financial metrics highlight a shift in rules of the game for the designers. The focus for the startup is therefore firm performance and survival, which may or may not be at odds with what designers prioritize in large software design teams. Further research is needed to understand how changes in these rules affect attention allocation and design in startups.

Table 2-1: Performance Metrics for Designers in Software Startups

Game	**Metrics**
Software Design Game	
Quality	Maintainability, flexibility, testability, correctness, reliability, efficiency, integrity, usability, portability, reusability, and interoperability (McCall, Richards, & Walters, 1977)
Project management	Budget, schedule, time to market (e.g., Highsmith, 2002; Molnar & Nandhakumar, 2009)
Digital Business Design Game	
Startup growth	Burn rate, activation rate, daily to monthly active users' ratio, customer churn rate, revenue growth rate, net promoter score (e.g., David Ehrenberg, 2014)
Funding milestones	Seed: Initial market research, product to build, initial market Series A: Figure out user base, product Series B: Software that can scale, build business Series C: More scaling, e.g., international (Delventhal, 2017)

Overall, the interactions between the components of the attention structures, channels, and environment are black boxed. There remains little understanding of how the different elements of attention structures and contextual factors concurrently influence the attention allocation of decision-makers between conflicting organizational goals (Stevens, Moray, Bruneel, & Clarysse, 2015). Instead, the ABV assumes that the attention structures and procedural and communication channels will lead to the convergence of attention allocation within the different units within the firm. Designers may also respond differently from the ways in which "a firm's attention structures and channels distribute attention to him or her" (Barnett, 2008, p. 611). How these designers allocate their attention can affect the design alternatives they choose.

2.6 SUMMARY

In this chapter, I begin with an overview of the entrepreneurial context and how its implication for the work designers do. I also briefly describe the existing software design research and the attention-based view (ABV). Analysis of the software design research

using the ABV highlights gaps in the software design research that have emerged because of the entrepreneurial context, particularly with respect to the players and rules of the game. Overall, the dynamics of the conflict resulting from the interactions between the components of the attention structures, channels, and environment are black boxed. Past research treated the processes and outcomes of entrepreneurship as distinct units of analysis and either focused on one or the other (process or outcome), whilst ignoring the interdependencies between both (Nambisan et al., 2017).

Chapter 3: Site and Methods

3.1 Overview

This study strives to present a richly detailed account of how the entrepreneurial context and players affect design decisions and design products. I have adopted a qualitative, inductive approach to study the work of the firm's designers at HealthCom. The case study strategy is appropriate for use in research when context must be explored to understand a phenomenon (Yin, 1994) and the "dynamics present within the setting" (Eisenhardt, 1989, p. 534). Case study research has often been used in the study of how work is done (e.g. Bailey, Leonardi, & Barley, 2012; Tumbas, Seidel, Berente, & Brocke, 2015; Antonopoulou, Nandhakumar, & Henfridsson, 2016). Entrepreneurship researchers have pointed to local context as an essential moderator of how entrepreneurial work is the context, the dynamics, and the organizational factors that play into design (Zahra, 2007; Zahra et al., 2014). Due to the exploratory nature of this study, the participant–observation case study was adopted to be able to understand real-time the effect of context on design process and product.

Multiple data sources exist in the use of the single case study. The methods of data collection included: semi structured interviews, ethnographic observation, and document review, which allowed for the triangulation of data. Particularly critical to this analysis was the corpus of chat transcripts that held a record of interactions between designers, even prior to my entry into the field. Iterative qualitative coding was used in my data analysis. With an extensive corpus of data suitable for constructing and confirming a holistic view of design decisions and design work, this case study brings together data and analytical insights at the levels of individuals and the firm to inform research on design work. In this dissertation, I use an interpretive process to construct theory that is grounded in the data (Walsham, 1995). At the same time, the analysis conducted is also informed by the existing

literature on software design, entrepreneurship, and attention-based view (ABV) literature. I started my analysis by adopting the designers' terms and meanings in the themes that emerge from the data. I also attempted to disengage from the existing theoretical work on software design in the analysis, keeping in mind only the key practices in software design, to facilitate the emergence of meaning from the data.

Although the data sources are described separately from the procedures for data analysis, concurrent data collection and analysis occurred throughout the duration of the study, as recommended by grounded theory researchers (Charmaz & Mitchell, 2001; Emerson, Fretz, & Shaw, 2001; Strauss & Corbin, 1998). The insights from early and ongoing data analysis have guided my data collection activities, since they suggest areas for additional observation and interviews. For example, themes that emerged later prompted follow up interviews and conversations for further examination.

In the remainder of the chapter, I provide background information on the research site and software designed and the procedures for data collection and analysis.

3.2 CASE BACKGROUND

This section provides a description of HealthCom, the software designed, and the design process. I also briefly summarize the industry level initiatives and the clients of the firm that prompted the design of this software.

3.2.1 The Research Site: HealthCom

HealthCom (a pseudonym), a digital health startup in the southern United States, was selected as the case study site. At the point of entry into the firm, the firm was 16 months old. The study terminated in Month 34. The firm was also acquired in Month 42.

HealthCom was particularly well-suited for my research question for several reasons. HealthCom's co-founders faced unknown, emergent requirements, as the software

and the digital business grew. The healthcare context is particularly challenging to design for: knowledge transfer in healthcare involves complex treatment plans and users of varying socioeconomic status and physiological conditions (Lim, Jarvenpaa, & Lanham, 2015). Healthcare clients are also slow adopters of new innovations (Lim & Anderson, 2016). Slow adoption and long lead times during contract negotiation were at odds with the contracts and revenue needed for startup survival. The entrepreneurs also had no background in healthcare. They were entrepreneurs who believed that they could change the healthcare industry and design a different software. Therefore, it was intriguing to examine how they would approach design differently.

The firm also provided access to the firm's design process in the early years that was not easily apparent to outsiders. It allowed me to observe the unfolding of the design process and product in real time, over a period of 18 months, without ex-poste knowledge or biases about the success or failure of the firm. Being able to observe the work of the designers in close detail over time, and from multiple perspectives, allowed for a deeper understanding and interpretation of the data sources, such as the chat transcripts. Access to the chat transcripts also allowed me to capture conversations that occurred when I was not at the firm. Access to the phenomenon at such a fine-grained level would not be possible in large sample statistical analysis or even through the common method of examining letters to shareholders in ABV research. The single case was thus ideal for observing the micro-processes in daily work and interactions between the individuals, organization, and the environment.

3.2.2 The HealthCom designers

In this study, I called all firm employees designers. They all contributed to the design of the software and the digital business. All the designers, even the Customer

Liaison, had computer programming and design backgrounds and were involved in discussions that shaped the design of the software. They also had experience in the entrepreneurial setting, either as owners or participants in other startups. All the designers, except for the Customer Liaison, were male. In contrast, most of the users and the intermediaries they dealt with were female.

HealthCom was founded by the CEO and Chief Designer (the co-founders) in 2013. The co-founders both had previous successful entrepreneurial firms. They started HealthCom with a different business model and software idea. They changed their idea after participating in an accelerator and meeting their first two clients, who were also investors (Clients A and C). They then recruited the CTO and iOS designer (January 2014) in order to create the first version of the software. During this time, they also hired contract staff to assist with the design of components of the software, e.g., the web dashboard.

They later hired a designer to work on the web dashboard and database architecture (i.e., Designer P), since working with contractors on a part time basis meant less consistency in the design process. In July 2014, they hired an Android designer and an intern (myself). A Customer Liaison was added in November 2014 to manage the users' requests and help with training. In 2015, they hired another designer (Platform Designer L) and replaced designers within the firm who left. At the termination of my study, they had a total of nine employees. There was also staff turnover during this period, whose input was included in my analysis. Thus, the data corpus includes quotes from more than nine designers. To protect my participants, I used pseudonyms or roles for the designers, clients, and the firm. An overview of the HealthCom designers is provided in Table 3-1.

Table 3-1: Overview of HealthCom Designers

Role	Designer	Background	Tenure	Nature of work	No. of speaking turns
co-founders	CEO	Acquired previous startups. Designed other consumer apps. First time in healthcare. Trained in computer science and business.	Mar. 2013- end of study	Started by creating the prototype and coding. Moved to QA and then finally focused on selling the software.	1747
	Chief Designer	Worked and held leadership position in several startups, some of which were acquired. Focus on product design. Trained in business.	Mar. 2013- end of study	Managed company operations and overall product design.	6404
	CTO	Involved with previous startups. Acquired three startups. Trained in computer science	Oct. 2013- end of study	Managed data architecture, oversight of all technical design.	4949
	Customer Liaison	Worked with other startups in account management, with customer facing role. Has an MBA. Also founded own startup.	Nov. 2014- end of study	Interacted with users (limited) and intermediaries at the client sites. Translated client requirements into design requirements.	509
Mobile team	iOS Designer	Previous employee of CEO. Involved in previous startups. Computer Science major.	Jan. 2014- Oct. 2015	Designed the initial Android app and entire iOS app. Designed APIs for the mobile and web interfaces. Liaised with the Chief Designer for UI related decisions.	5138
	Android Designer	Previous employee of CEO. Computer Science major.	Jul. 2014- Oct. 2015	Responsible for the Android app. Liaised with the Chief Designer for the UI related decisions. Liaised with the iOS designer for mobile platform (iOS /Android) related design.	954
Platform Designer	Platform Designer L	Software developer for several companies. Previous involvement with six startups.	Nov.2014- Apr. 2015	Member of the software design team, responsible for API design and integration with other tools used in design (e.g., Mandrill). Reported to the CTO.	389
	Platform Designer C	Replaced Platform Designer L. Computer science major - worked in several startups as programmer.	June 2015	Responsible for platform and API development and integration with other tools used in design. Reported to the CTO.	634
Web Designer	Designer P	Worked on web/ database technologies	May 2014- Oct. 2014	Continued design of web dashboard. Supported database design (assisted CTO). Reported to the CTO.	703
	Web Designer O	Part of same startup accelerator as HealthCom co-founders.	Jan. 2014 - end of study	Took over the design of web dashboard from Designer P.	1356

The first column in Table 3-1 highlights individual's roles throughout the duration of the study. For instance, Web Designer O took over from Designer P on the design of the web dashboard. Conversely, Platform Designer C took over Platform Designer L's design work.

3.2.3 The HealthCom software

HealthCom created the software to facilitate communication and knowledge transfer between nurses and patients in the pre-or post-treatment phases. The HealthCom software included a backend database on the server, iOS and Android mobile applications, and web portals that must be coordinated in the design process.

Figure 3-1: Simplified Representation of Software Design

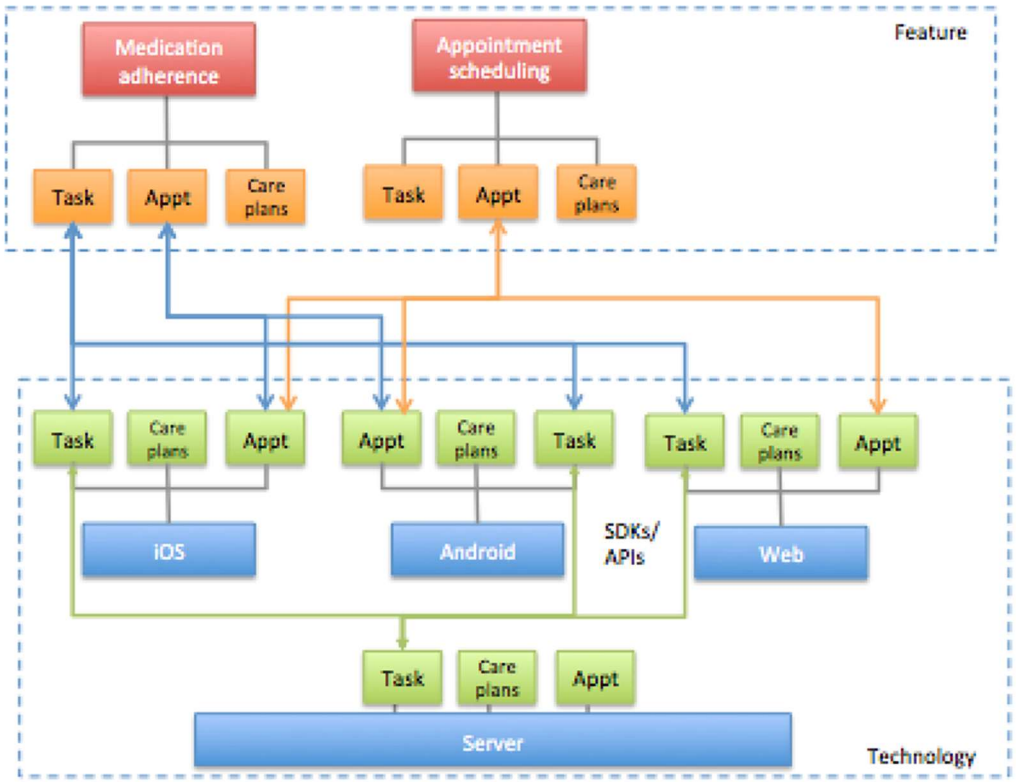


Figure 3-1 is a simplified representation of the software design. The iOS, Android, Web, and Platform Designer and the CTO worked together to create the different components of the software. For example, an appointment setting feature had to be made available on the iOS, Android, and web platforms. There were also interdependencies between features. For example, if the patient had surgery on January 1st, care instructions would be populated on the software for pre-and post-surgery on January 1st. Thus, the tasks and the care plan (general list of tasks) were dependent on the functioning of the appointment (labeled *appt* in Figure 3-1) feature. Coordination between designers was, therefore, important for the creation of any feature.

Table 3-2: List of Features Designed

	Features	**Description**
1	**Share my care**	Patients can give their family and friends permission to view the care plan assigned by the provider and completion of tasks.
2	**SecuredMessaging**	Gives patients the ability to send their providers a secured message from the mobile app or patient web portal.
3	**Gamification**	Incorporates elements of game playing into the care plan (such as stars for completing tasks)
4	**SDK**	Software development kit (SDK): tools that allow the creation of applications from the HealthCom infrastructure.
5	**Medication Adherence**	Allows patients to manage their list of medications.
6	**Clinical content**	(1) Education content that patients can reference on the infrastructure about their medical condition and (2) care plan tasks
7	**Reporting**	Summary of all tasks completed.
8	**GPS-enabled login**	Nurses/aides can log their visit through the app for reimbursement from insurance payors or state funding. Relevant for certain clients
9	**Data analytics**	The ability for patients and caregivers to view trends about their care plan task completion rates, most commonly reported items, etc.
10	**Web portal (nurse)**	Allows nurses to create a care plan, upload educational content and monitor patients.
11	**Web portal (patient)**	Permits patients to complete the tasks assigned and to track their progress.
12	**Patient Onboarding**	Started with manual uploading of patient data and the creation of patient accounts. Kiosks were then used to help the onboarding process.
13	**Appointment**	Allows patients to log their appointment date so that the tasks can be automatically populated. The appointment feature would ideally integrate with the client's scheduling software.

Table 3-2: List of features designed (continued)

	Features	Description
14	**Care Center**	Hosts educational content and clinic information. Expansion into different content types (e.g. videos)
15	**Survey**	Questionnaire to allow for clinical decision support or more complex reporting of health status
16	**Tasks /Care Plan**	Tasks that patients received. Expansion into different tasks types. The Care Plan comprises a list of tasks that the patients had to complete.
17	**Sounds**	Sound notification to alert the user about a specific task / completion of a task
18	**Alerts**	To alert the nurse or provider when a patient is non-compliant with the task or not performing as expected
19	**Signature**	Capture signature on care plan, either by nurse or by patient (receipt of services)
20	**Payment checkout**	To allow patients to pay for the app services (only for Prospective Client K)

The list of features designed by HealthCom during the three-year period are summarized in Table 3-2. These features will be referenced throughout the dissertation. They were either designed as separate features or were extensions of existing features.

Figure 3-2: Co-Evolution of Business Opportunities and Features Designed

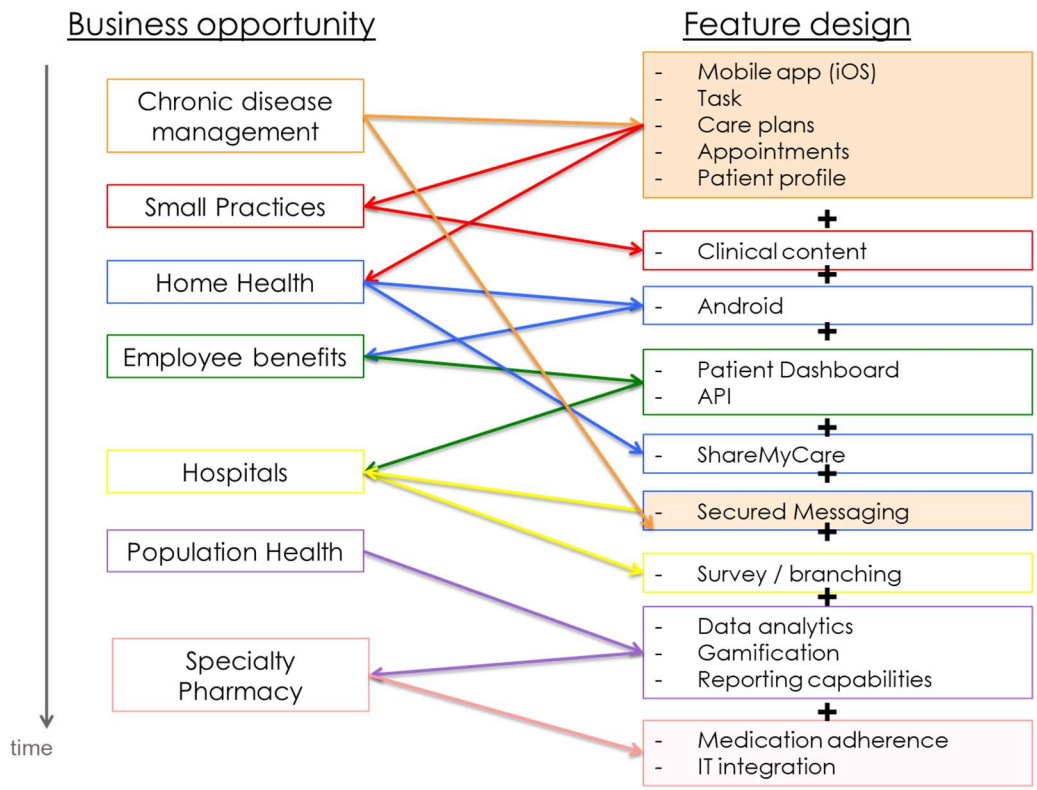


Figure 3-2 summarizes how the features designed were a result of HealthCom trying to capture emerging business opportunities. Instead of working with a product vision and road map of features that was designer directed, the features to be designed were dictated by the business opportunities that emerged for the firm. The left axis refers to time. The boxes on the left indicate the business opportunities that were pursued by the HealthCom co-founders. The HealthCom software was not a disease specific software. Therefore, the co-founders were exploratory in their search for clients. Every client that the firm signed on also requested for new features to be added to the software. As a result, the software design evolution was dictated by the requirements of the prospective clients. The features that were designed would then be made available to all clients of the firm, since the software was on the cloud. The software was implemented at its first client site in 2014. HealthCom then released updated versions of the software every four to six weeks. These versions included new features or solutions for defects that would be available to all clients.

3.2.4 HealthCom's Clients

HealthCom licensed the health IT software to clients (hereafter referred to as "contract"), such as a home health agency and a population health management company. While these clients were all in the healthcare industry, they had very different use cases and feature requests (see Table 3-3). Note that Clients M, T, and H were "pending," since they were still in negotiation and were not confirmed as clients at the end of the study.

Table 3-3: List of HealthCom's Clients

Client Name	**Area**	**Use for software**
Client C	Infusion services	Used for treatment monitoring
Client A	Home Health	Used by nurse aides as a to-do list when visiting patients' homes
Client B	Employee Benefits	Used by case managers
Client D	Dialysis (terminated)	Used for treatment monitoring
Client E	Health IT	Used as a communication tool with users
Pending at the termination of study:		
Client P	Specialty Pharmacy	Used for treatment monitoring
Client K	Disease management	Used for treatment monitoring
Client M	Academic medical hospital	Communication between case managers and patients
Client T	Pharmaceutical company	Used as a tool for recording treatment administration
Client H	Patient Education	Communication between case managers and patients

The reasons for adopting the software varied by client. Some clients, such as Client A and Client T, used the HealthCom software as a to-do list for the nurses providing treatment to patients in their homes. The HealthCom software was, therefore, viewed as a business process improvement tool. In another case, Client C was a small specialty treatment center that sought to use the software to improve patient satisfaction. In all of the cases, the client paid for the licensing of the software and had intermediaries involved in the negotiation process for the requirement design. The designers, therefore, had to manage the different requests and revenue targets.

Business opportunities also arose because of macro-level policy initiatives. The Affordable Care Act and other reimbursement policies implemented by the United States government and Centers for Medicare and Medicaid, respectively, created incentives for healthcare providers in the United States to reduce patient readmissions and improve patient satisfaction. There was a need for increased communication and supporting software between patients and nurses to reduce complications that patients encountered post treatment. These incentives, that were created prior to and over the duration of the study, motivated clients to license a software like HealthCom.

3.2.5 HealthCom's Design Process

Request for features were informed by prospective or existing clients, such as through marketing presentations, Requests were received by the CEO, Chief Designer, or Customer Liaison who interacted with client intermediaries on a regular basis. The Chief Designer would then create the overall user interface and supporting graphics for the software. The other designers would execute the code necessary to operationalize the feature. For example, the iOS designer would create the iOS mobile application, the Web Designer was responsible for the web portal, and the platform designer and CTO developed the supporting database and APIs for the feature to function across all the platforms. The Customer Liaison would then manage the clients by addressing the clients' ongoing requests in weekly calls or by providing training to users.

The designers interacted through emails, in-person conversations, online project management boards, and most commonly, an online chatroom. The Android and iOS designers were located offsite for most of the study. Twice a week, there would be a video conference call to coordinate the requirements and deadlines for the design product and to seek feedback on designs.

At each client site, access to end users (such as nurses or patients) was also limited. HealthCom's designers provided training prior to the deployment of the software at each client site. In some of the sites, the HealthCom designers could interact with users. However, such opportunities were rare. The designers (co-founders, Customer Liaison) mostly interacted with client intermediaries. Feedback about software use was also not collected nor analyzed. Any client feedback in the design occurred early in the design process, when the Chief Designer or Customer Liaison circulated screenshots of planned designs to new clients. Otherwise, feedback came through the reporting of defects or complaints after the feature had been implemented during the weekly calls between the

Customer Liaison and client intermediaries. Clients intermediaries also did not always provide detailed requirements for the design. The prospective client dictated the requirements, albeit at an abstract level, even though the feature designed would be made available to all clients.

3.3 Preparing for the Field

One of the challenges of starting a qualitative project is developing the base knowledge necessary to understand what the participants are doing. Software design occurs in a collaborative fashion involving multiple players that have a claim on the desired features (Lyytinen, 1988). To understand the work of the HealthCom designers meant to understand not only the software design but also the broader context within which requirements are navigated.

Keeping these findings in mind, I believed that it was important for me to develop a basic familiarity with this world before entering the field. To understand the entrepreneurial context, I sought to understand the constraints of HealthCom and other health IT firms by attending professional conferences and talking to entrepreneurs, clients, investors, and industry experts. My goal was to better understand the challenges that these startups could face in order to triangulate my understanding of the health IT entrepreneurial environment. Through conversations with the different players, I developed an appreciation for the challenges and decisions that entrepreneurs had to deal with as they designed the software. I also read books recommended by entrepreneurs, such as Lean Startup, to understand the practices entrepreneurs were likely to adopt.

The biggest missing piece of my knowledge was software design. I asked the CTO for primer books to understand basic design heuristics often employed in software design. My lack of software design knowledge was the my most daunting barrier. While it was

impossible to learn software design prior to field entry, my goal was to develop an initial level of what Collins and Evans (2007) call "interactional expertise," defined as "expertise in the language of a specialism in the absence of expertise in its practice" (p. 28). Familiarity with software design was critical for understanding the nature of the field work. With a rudimentary understanding of software, I felt prepared to engage with HealthCom. I also continuously learned throughout my observations, by relying on the Internet to look up technical terms that were used in conversations. Not only did the preparation provide me with a basic understanding of the world that I was entering, but I also soon discovered that my demonstrated interest and perceived ignorance, led designers to be more thorough in explaining the rationale behind their design decisions.

3.3.1 Identifying a Site

Beginning in the summer of 2014, I initiated contact with founders of digital health startups all around the United States, especially those in my professional network. I corresponded with these firms to express my interest in learning about their design process and let them know about my interest in conducting dissertation research. I talked to many organizations across the United States.

My initial contact with HealthCom occurred in May 2014 during discussions about another research project. As I talked to the client, it became clear that HealthCom was an ideal research site to examine process of design. It would provide me the ability to observe the design process in real time. I pitched the possibility of using HealthCom as my field site, and my contact was open to facilitating my access.

In exchange for access to the site, I was to be a paid intern responsible for generating user reports for the clients and providing insight into possible client requirements for the analytics feature. The CEO initially wanted me to do marketing

presentations. I declined because it would lead me to become too involved in the design or strategic process. Generating reports was an isolated task that allowed me access to the office and to the internal communications, while not affecting design or strategic decisions. As a startup, they were resource constrained and needed help with the mundane work that was taking time away from design. I was an easy sell, since all I needed was a space to work and had healthcare knowledge that they could potentially leverage. I had been a hospital administrator for a major academic center and had been on the client end, licensing software from vendors. I had also been involved in the design of physician order software and had over 1000 hours of nurse and patient observations. I was, therefore, very familiar with the healthcare context, even more than the HealthCom founders, who had no prior healthcare experience. They ultimately did not leverage my healthcare knowledge because the designers were often inattentive to the needs of the users, as the findings later would show.

During my time at HealthCom, I had a dedicated work space in the open office. Having this space permitted me to listen to many different conversations. Because of my work, I was allowed access to chatrooms, internal emails, and meetings, without having to provide input into the discussion process. Since I was collocated with the other participants, I was able to be continuously aware of their activity.

The designers were aware of my dual identity as an intern and researcher. It also gave me access to the clients (e.g., client visits) and connections to clients and industry experts. My role as a PhD student also afforded confidentiality, as the informants were candid in their evaluations of the HealthCom software. Being a member of the staff and chat room also helped fill in the gaps for the times when I was not on site, since I could review the history for events discussed.

3.4 Methods of Data Collection

My data collection involved three techniques: review of the archive of digital communication records, ethnographic observations, and semi-structured interviews. My data collection strategy focused on understanding the designers' interactions, design rationales, and actions. Data collection was informed by my greater interest: to understand how they responded to demands of the entrepreneurial context at the firm and individual levels and the impact on the design product.

Data collection occurred over an 18-month period from July 2014 to December 2015. The primary set of data included: (1) chat transcripts, 1956 pages, among the designers in an online chat room, stemming from the firm's early months (nine months old), (2) design tasks, totaling 2,370, from an online project management tool, (3) emails, 1412, exchanged, and (4) access to the founders' calendars. The archive of digital communication was triangulated by (a) semi-structured interviews, 21 in number, with the employees and founders, its clients, investors, and industry experts, (b) field notes, and (c) internal documents and public information about the firm. Later, as the firm design process changed, they also started including more user stories on the project management board that could be used to understand design rationales. Table 3-4 and Table 3-5 summarize the data sources.

Table 3-4: Data Sources

Data source	**Quantity**	
Interviews		See Table 3-5
Internal Emails	1,412 (17 months)	All employees
Chat transcripts	1,956 pages (24 months)	All employees
Observational data	17 months (3–6 hours per week)	Notes from weekly product meetings and standup meetings; discussions of conversations in the office
Project management board	2,371 tasks listed	All employees
Client site visit: interactions with nurses, administration, and patients	1 site visit	Author's notes from visit to HealthCom's first client
Presentations (e.g., marketing presentations)	3	Author's notes from attendance at presentations
Strategic report to board and investors	2	CEO
CEO's calendar	Record of every meeting since the founding of the firm	

Use of data from several sources and hierarchies within and outside of the firm provided me with a deeper understanding of the phenomenon.

3.4.1 Archival Data: Chat Transcripts and Emails

The first and most critical method in my corpus of data was the chat room transcripts and other archival records of digital communication. The online chat room was the key form of communication for the designers. The availability of chat transcripts preceded the start of this study.

Discussions between designers often occurred in the online chat room. When I first entered the firm, I spent my first three hours familiarizing myself with the software and realizing that no one was speaking to each other. Other than an occasional laugh or brief conversation, there was not a lot of in-person conversations. I was not sure what I would or could be observing in the startup. After mentioning my confusion to the Chief Designer, he clarified that most discussions occurred in an online chat room, which he then gave me access to.

The designers worked in a work-sharing open office setting in the initial months of the study. Because of health information privacy concerns, the firm's designers were cautious about discussing user information in the open setting. Any team discussion would occur in a meeting room. Two months later, the firm moved into a private office that gave them more space to hire designers. Conversations also became more candid. However, due to the openness of the office space, the co-founders cautioned that discussions should be held in private rooms or on the online chatroom. The chat room continued to be an invaluable archive of digital communication within the firm, although I did not have access to private messages, private emails, or all in-person discussions.

Analysis of conversations allows researchers to generate insights that may otherwise be unobservable and neglected (Kyprianou, Graebner & Rindova, 2015). The ability to use archival communication data thus allowed for convenient recursive iteration between theory and data for process theory building or emerging ideas in the theorized process (Levina & Vaast, 2016)

Other forms of document reviews included internal emails (on which I was included), project management boards, access marketing material, publicly available information, such as blog posts on websites, or twitter feeds, etc. My awareness of these documents occurred mostly through ethnographic observation and/or availability through the chatroom or emails. I had access to most of these, although there were some things, such as business plans, that the co-founders were reluctant to provide me access to. They were also wary of giving me too much access because I was not an employee of the firm and felt that there was no reason to share the information.

I also collected screenshots and supplemented the description of specific activities within the field notes from ethnographic observation and discussions around the screenshots. Links to screenshots were often embedded within the chatroom transcript. I,

therefore, made notes next to related data when they appeared in the transcripts. The ability to analyze the data after the study was also subject to my ability to export the archive of communication: for instance, I could not export the comments from tasks on the project management board online, and data collection was limited to my notes from my earlier readings of the comments. I included all of these documents in my ATLAS.ti project. I also maintained them separately from field notes and interview transcripts in my file management.

3.4.2 Ethnographic Observations

My second source of data was ethnographic observation. For the duration of my time at HealthCom, I spent three to six hours each week (depending on my class schedule), visiting on different days, if possible. Other activities, such as visits to client sites or meeting with industry experts, also required some data collection outside of the established schedule.

Ethnographic observation was important because it helped me make sense of the chat transcripts or observe first-hand the concurrent online/offline discussions that may occur. For instance, because the iOS designer was working offsite, heated debates between the Chief Designer and iOS designer could be occurring concurrently online, with additional comments from the Chief Designer offline. The structured interviews and document review was greatly augmented by numerous informal interactions afforded me due to my co-location with the designers. For example, as I wrote up my field notes in the days following an observation, I was able to approach the designers for clarifying questions. As I got to know the designers, I was invited to many informal events, such as coffee, meals, or happy hours. Even though I was not taking field notes during these moments, the conversations that occurred in these spaces helped me to develop a more

nuanced understanding of the designers' perspectives that, in turn, helped me interpret my data in a situated manner.

Because the designers were busy during the workday, I would take the chance to speak with them individually at lunch or for short periods throughout the day. Following up frequently also ensured that the event was more salient in their mind. Most of these informal conversations were not recorded, though for longer interviews oral consent was obtained per the IRB. The informal conversations were typed up as part of my field notes after returning to my desk. I summarized my recollection of the interaction and my thoughts about why it was relevant to my research question. There were also times when I would tell the informant that I need to record the information on the notes application of my cellphone because it was a very interesting or insightful point. Due to the time constraints and open office layout of the firm, it was difficult to ask participants to maintain a running monologue of their actions. Therefore, I often asked questions about design decisions or behaviors after reading about conversations in chatrooms (while getting coffee) to clarify my understanding.

On some days, observations were limited. There were few in-person conversations and designers were hunched over their computers. At other times, observations were richer, such as when I attended meetings or overheard conversations between designers. Following each observation, I summarized that day's activity into a running account of participant work activity. I typed up my field notes integrating my observations of the firm.

3.4.3 Semi-Structured Interviews

Semi structured interviews first occurred with the co-founders at the start of the study, and then with the other designers throughout the study and at the end of the study.

Several interviews were also conducted with individuals when new insights or themes were identified in my data (see Table 3-5).

Table 3-5: Informants

	Informant	Date interviewed	Pages
Designers			
1-2	Chief Executive Officer (CEO)	March 2015, April 2016	1.5 hours total
3-4	Chief designer	May 2014, October 2015	1.5 hours total
5-6	Chief Technical Officer (CTO)	October 2015, Mar 2017	1 hour total
7	Customer Liaison	December 2015	45 minutes
8	Platform Designer L	October 2015	45 minutes
9	iOS designer	October 2015	45 minutes
10	Android designer	January 2017	45 minutes
11	Group interview (engineers, Chief Designer, Customer Liaison)	April 2015	1 hour
Non-designers			
1-2	Interviews with prospective clients that did not materialize: Client V and Client Y	Feb 2016, May 2016	2 hours
3	Liaison at the business accelerator	March 2016	30 minutes
4-6	Business consultants hired: Consultant W, Consultant N, Consultant M	Feb 2016, April 2016, May 2016	2.5 hours
7	Industry expert	Feb 2016	1.5 hours
8-10	Investors: Investor S, Investor R, Investor L	May 2016, June 2016	1.75 hours

I collected 11 interviews from the designers and 10 interviews from non-designers (See Table 3-5). Each interview lasted between 30 minutes and 1.5 hours. The average length of each interview was approximately 46 minutes. These counts reflect limitations on the number of informants who could speak with me about the design process, given the small size of the firm. Because I wished to understand the views of a broad variety of players in the design process to answer my research questions, I selected semi-structured qualitative interviewing as a data collection technique. Semi-structured interviewing refers to the use of guiding instead of strict protocol for the interview.

I developed the questions in the semi-structured interview protocol to prompt informants to speak about their experiences in detail during interviews. I asked the

designers to reflect in detail about their work and particularly their designs, their rationale, and the challenges of the design. I chose a semi-structured format because it permitted a dialog with my participants that would allow me the freedom to probe topics as they emerged and because it afforded my participants the choice of identifying potential topics that I, as an outsider, would not have known to ask about (Kvale & Brinkmann, 2009) In other interviews, such as with the CEO, I focused more on the strategy business and how that influenced the design. I adjusted the interview protocols based on the role of the individual designer. For instance, I asked the Android designer about decisions about medication adherence but also the interdependencies between the designers. I asked the Customer Liaison about the design requests from clients and how they were prioritized. Although the general structure of the protocol was consistent across the interviews, I continually modified my questions based on my observations and analysis of the data. Modification of questions is consistent with an iterative and inductive development of grounded theory (Glaser, 1978; Strauss & Corbin, 1998).

Due to the designer's time constraints, my interviews lasted between 30 to 45 minutes. During the interviews, I took limited notes to inspire probing questions and record major points of interest. I then integrated my interview notes with the actual transcripts to develop expanded transcripts (Spradley, 1979).

As a newcomer to software design, I used the interviews to develop a basic understanding of the software design process. In each interview, I asked a series of questions focused on the processes, rationales, and challenges. Informants frequently referred me to resources from websites, books, or online forums. To get participants to speak less generally, I probed for specific recent examples from their work. As I performed more observations, I brought examples observed into the discussion as a means of obtaining my participants' perceptions of their interactions. For instance, in my earlier

interviews, I asked questions about how design in a business to business or healthcare context differed from their previous experiences in consumer-based applications. I asked participants to describe situations where they faced tensions or disagreements and how they were reconciled. In addition to addressing these specific questions, some of these follow-up discussions also served to broaden my understanding of aspects of design process, obtaining a final narrative of how the design occurred.

It was also difficult to keep the interviews structured, since they started to feel unnatural to both the designers and myself over time, as I was interacting with these designers regularly and at social events. I kept broadly to the protocol by trying to include some of these questions into other questions as clarification to check my understanding. For example, I would reference certain events with informants or start with my question about the software design, beginning with my basic understanding as a catalyst for the conversation. This approach allowed me to elicit specific examples from informants.

My frequent interactions with the designers sometimes posed challenges in the interview process. Even though my intention was to elicit the designers' response in their own words, the designers would look confused and say, "as you already know." In response, I would briefly describe my understanding but not provide my personal views of the events. The designers did not hesitate to correct me if they disagreed with my understanding. I always tried to play the ignorant observer and ask why they chose to do what they did if it contradicted the practices espoused by the practitioner or academic literature. One limitation was that I could guide informants down certain paths that were not intended. To mitigate this, I would often follow up my questions with "what else" questions.

I began talking to other clients, industry experts, etc., based on the snowballing technique and personal connections. In some cases, the co-founders directed me to other

individuals they thought could provide additional insight into the work they did. I also sought out informants from personal connections to get their insights about HealthCom. For instance, I realized after a marketing presentation that they were trying to sell the software to my ex-colleague. I followed up with my ex-colleague and got candid feedback about the software and HealthCom. My prior healthcare experience thus facilitated many of these interviews. My strategy was to try to talk to other players mentioned by HealthCom to get their perspectives. For the other informants (non-designers), I adhered more strictly to the protocol.

The goal of generating an extensive corpus was to capture enough information that the data reached theoretical saturation (Corbin & Strauss, 2008). Nonetheless, the case had problems of left and right censoring (Miles & Huberman, 1994). While I have information about the early months (prior to the start of the study) through chat transcripts, I was not there to observe the interactions. I tried to follow up through informal conversations with designers. The actual design of the HealthCom software begun about seven months prior to my arrival. In terms of right censoring, during the study the data I collected reached a point where additional observation failed to reveal new perspectives. I was not getting information that would have substantially affected the results of analysis. At the same time, the co-founders wanted me to be more involved in the operations of the firm, which I could not. I, therefore, decided to terminate my intern relationship, while agreeing to continue attending weekly meetings. A week later, they removed my access to all internal communication records, as the firm's lawyer was concerned about access to information without any employment relationship. I did not know that things would change, since the firm was acquired eight months after the termination of my study. The termination of my study prior to acquisition has implications for the boundary conditions of my findings.

3.5 Protecting Participants' Identities and Rights

As notified by the University of Texas at Austin's IRB, all designers and clients were notified about my role as a researcher through an email from the Chief Designer to the firm and clients. Verbal consent was obtained from all my participants. I was open with my participants about the level of anonymity that I would use in the write-up of this dissertation. I used pseudonyms for the firm and their identities. I anonymized all interview transcripts and field notes from observations to ensure that identifying information (primarily names) of informants did not appear in the data for analysis. Anonymization was impossible for email or other public documents for which identifying information was available on the document or if I could only export the data in a PDF format. All documents were stored in a password protected folder on my computer.

3.6 Data Analysis

Data collection and analysis are iterative steps in fieldwork. Ongoing analysis provided additional insights about how to proceed with the study. For example, I wrote up interim case reports, produced data displays, and kept records of my interpretations of events (Miles & Huberman, 1994). I also wrote several case summaries of individual designers and features (Miles and Huberman, 1994)

In this section, I refer primarily to the analysis activities that I conducted that culminated in the two findings chapters. I analyzed my data using techniques inspired by grounded theory (Glaser, 1978; Strauss & Corbin, 1998). My analysis began in an iterative fashion that started while I was in the field. Throughout my time in the field, I constantly reflected on the themes that emerged. These were written up in the form of conference submissions, class papers, or departmental presentations that helped me articulate my thoughts about the emerging themes.

These analyses were performed primarily using the ATLAS.ti qualitative analysis software. I also revisited the raw data frequently to refresh my memory and to reinterpret the data in terms of emerging themes. The first analysis, presented in Chapter 4, sought to understand how the environmental context affected the design action and priorities of the firm. The analysis highlighted the other players (client intermediaries and investors) that were the drivers in the discussion. Delving deeper, I looked at the rationales and considerations in each design discussion and emerged with the attention-based view (ABV) as an organizing framework. In Chapter 5, the findings highlighted different design actions and the impact on the design product. I will describe the analysis in greater detail in the following sections.

A preliminary round of reading interview transcripts and observation field notes sensitized me to the notion that minimality of effort in design figured heavily in the data and merited further investigation. As I performed my observations, read and re-read the chat transcripts, and collected additional interviews, I was consistently reflecting upon the initial patterns that emerged. For example, my observation that designers were often arguing about the minimum requirements for design led me to pursue various lines of inquiry that served as the basis for the analysis in the initial stages of open coding. Codes included codes on design actions (e.g., "limit scope of design," "extending function"), actors (who was in the conversation), the designers' goals for the design ("good design practice," "user satisfaction"), etc. The idea of prioritization emerged in my line-by-line reading. In a subsequent pass of coding, I flagged mentions and instances of prioritization (e.g., "do it right now," "postpone," "postpone although easy fix") in design discussions, which led me to identify the determinants of acceptable performance levels (such as lowered quality thresholds) in the process.

With these preliminary analyses, insights emerged on the players whom designers were focused on and the influence of the entrepreneurial context in each design discussion. As a preliminary analysis, I wrote case summaries for each feature designed, isolated all related design discussions, and identified the origins of design, considerations, and sources of tension in the design. I also began redefining, conflating codes or splitting up codes to become more granular.

With these insights, I proceeded to examine more deeply the differences and similarities in the design process and the implications for the design product. Subsequent analyses returned to the research question identified to investigate how the environmental factors influenced the design process and product. I focused first on interactions that occurred within the firm and how they were driven by environmental factors (such as the players and the firm performance). I also examined the impact of attention to financial-focused design alternatives on the collective design actions and the design product.

To follow up on the preliminary analyses, I proceeded to look closely at instances of particular themes in the data. These included notions of minimality and prioritization as described earlier. In each design decision, I coded for what the designers focused on and advocated for (the "who," "what," "when," and "how") in the discussions and what the decision was. I returned to the corpus in search of more concrete examples of the design discussions, in which disagreements occurred. I started coding for players (including investors, clients, and users) and the perspectives the designers were taking (such as user satisfaction, software quality, speed of design, business focused) to determine their focus in design. I constantly compared across instances of coding to ensure that new instances of the code matched that of the previous use (Glaser, 1965). Examples that did not fit were marked and reexamined at the end of the coding exercise to see if it warranted an expansion

of the definition or the creation of a new code. The definitions were also summarized in a codebook that provides a definition of each code and when they were applied.

I used the multiple data sources to triangulate the factors that influenced design decisions. At this point, the attention-based view emerged as a potential theoretical framework, since the analysis was directed at: (1) what the designers were focused on in the discussion (issues, concerns, who), (2) the design alternatives considered, and (3) triggers that prompted the focus on those issues and alternatives. Driven by my theoretical framework, my analysis for Chapter 4 began by developing a structured understanding of how the attention drivers originating from the entrepreneurial context affected the designers' attention allocation. This resulted in the identification of two distinct attention drivers: financially focused versus quality focused that led to attention to different design alternatives. The salience of the financial-focused attention driver increased with pressure from investors and poor firm performance.

Driven by this distinction, I coded the conversations for the differences in perspective that were directed by the two attention drivers. I then mapped the design actions, focusing on the design attributes and design product that resulted under the two attention paths. I grouped the content of the design discussion into recurrent themes which directed attention to different attributes of quality. The tension between financial and quality focused attention drivers emerged as the driver of design decisions. I isolated three key tensions of inhibiting, compromising, and over-optimizing that designers had to balance in the design process, which often led to disagreements in design discussions. Inhibiting referred to the case in which the design decisions inhibited financial performance or future changes to the design. Attention to the creation of financial-focused designs amidst time constraints meant compromising quality of the software. Designers who focused on quality were viewed as over-optimizing quality unnecessarily. The chat

transcripts provided examples of the distinct perspectives of designers towards design, while interviews and field notes verified these perspectives and tensions and allowed me to understand in the designers' own words their interpretation of interactions.

One thing that stood out was how the design product differed from what is commonly understood to be quality software in software design. The second analysis, presented in Chapter 5, sought to provide insight into the design actions and how they culminated in the design product. I began by isolating every design action undertaken by the designer and categorizing these design actions. I also looked for evidence of design outcomes (such as redesign, defects, limited in scope minimal and speed) and categorized them into five key characteristics of the design product. I typically located such statements in the chat transcripts, field notes, and interview transcripts.

In these rounds of coding, I marked statements describing the design process, rationales for their decisions, and their design actions. I sorted the responses and created new categories of designers. I also conflated ones that had significant overlaps. I utilized coding logs and memos (Corbin & Strauss, 2008) in the process of developing and refining categories. In coding logs, I recorded my reflections for each coding session, including the codes I had worked with, their definitions at the time of the coding session, as well as puzzles that arose while I coded and directions for the next coding session.

3.7 VALIDITY

To improve the validity of the analysis (Creswell & Miller, 2000; Davis, Yin, & Davis, 2014; LeCompte & Goetz, 1982), I adopted the following steps: triangulation of data sources, use of thick, rich description, and ongoing documentation of emerging themes and reflections.

Firstly, I constantly reminded myself to triangulate my findings across my data sources. As mentioned previously, my key data sources included archives of digital communication, observations and interviews. Triangulating across these sources allowed internal validity of the case study (Yin, 1994). Interviews could be subjected to post-hoc rationalization, and by comparing against the chat transcripts or my observations, I was able to examine consistencies and discrepancies in the behavior of informants. By speaking to several designers, I was able to get a variety of perspectives of the same phenomenon, thus allowing me to understand the phenomenon holistically. Having multiple data sources for the same phenomenon increases the chances of finding disconfirming evidence and alternative explanations (Fetterman, 1998). It also allows for the possibility of follow-up questions. For example, in my initial conversations with the Chief Designer, he often provided accounts of how he was a strong advocate for the user. It was not until conversations with the other designers and with other players and observations, did I realize that he was less open to ideas from the users than he described.

I also often engaged in fact checking to clarify my understanding of the phenomenon as well as the themes that were emerging from my data. Fact checking allowed me to present my interpretation of the phenomenon to the participants in my study. Fact checking was done on a one-on-one basis, as the findings may not always reflect positively on the firm or certain team members. I asked the informants most directly involved to confirm interpretations and discuss my interpretations of events. Prior to the sessions, I would prepare a one-page summary, figure, or table that highlighted my early analytical insights and asked informants to help me fill in the gaps in my records. These fact-checking sessions lasted about 45 minutes to an hour and often sparked later conversations and correspondence with informants. Fact checking improved the validity of constructs and other theoretical devices used to analyze data (Yin, 1994; Creswell & Miller,

2000). I also presented some results of my analysis at various conferences with researchers and practitioners for feedback.

Secondly, I tried to incorporate thick description and detail in my field notes (Creswell & Miller, 2000). As mentioned previously, this was sometimes difficult, given the lack of activity going on in the office. I tried to include more detail about the tone of voice, gestures of the participant offline, and during heated online debates, to provide detail to make the setting and events plausible in the mind of the reader (Creswell & Miller, 2000). Being there helped me to witness how the response (such as a sigh, yelling, or laughter offline) can lead to different interpretations of the conversations in the chat transcripts. My frequent interactions with the designers helped me to understand the office culture, politics, and dynamics so that I could interpret the archival data appropriately and detect nuances in the meaning of the responses observed.

Finally, I constantly coded, wrote memos, and recorded thoughts to create an audit trail of my analysis and reflections. I kept a collection of short notes and figures that mapped how my views changed about the process. I also reflected upon biases that may have emerged in the process. All of these documents are kept in a folder to facilitate the review of the evolution of the analysis. I also went back to the raw data often to make sure that the initial codes were still representative of the meaning, as my interpretation of the phenomenon might have changed over time.

3.8 SUMMARY

The research methodology used in this dissertation was presented in this chapter. I adopted a qualitative, inductive approach to study the work of the firm's designers at HealthCom. Due to the exploratory nature of this study, the case study was adopted to understand in real-time the effect of the entrepreneurial context on the design process and

product. Data sources included semi-structured interviews, ethnographic observation, and archives of digital communication. Particularly critical to this analysis was the corpus of chat transcripts that held a record of daily interactions between designers. Iterative qualitative coding was used in my data analysis. Finally, to validate the results of the analyses, I adopted the following steps: triangulation of data sources, use of thick, rich description, and ongoing documentation of emerging themes and reflections.

Chapter 4: Early Stage Entrepreneurial Context and Attention in Design

In this chapter, I highlight how the early stage entrepreneurial context introduces new players and rules of the game for HealthCom designers. There is a pull for the designers' limited attention and resources between financial and quality-focused decisions designs. I further introduce tensions the designers face, specifically among design alternatives that may inhibit future growth, compromise quality standards, or over-optimize quality unnecessarily.

4.1 Attention Drivers Resulting From the Early Stage Entrepreneurial Context

While attention drivers were not mentioned explicitly as a construct, designers justified their design decisions based on demands of certain *players* or in attempts to abide by certain *rules of the game* (i.e. for software or digital business design). When talking with each other, and with me, HealthCom designers frequently cited two key attention drivers that influenced their perspectives in the design process. I named the attention drivers that emerged from my analysis: *financial-focused* and *software-quality focused attention drivers*, as summarized in Table 4.1. I distinguish between these two attention drivers by looking at the rationale explicated in the conversations. While higher quality software design could be related to financial-focused attention drivers in the long-run, they were not viewed as complementary in the short-term in the discussions analyzed.

Table 4.1: Overview of the attention drivers evident in design discussions

Attention drivers	**Rules of the game**	**Players**	**Desired outcomes**	**Impact on design action**	**Representative quotes**	**Codes**
Financial-focused	Design should increase revenue and funding for the firm	Investors, clients	Financial outcomes, marketable designs	Focused on user interface and user experience, Limited scope of design to what is necessary for the client	1. *I can confirm that [Prospective Client K] has expressed a need for this [feature] as well.* 2. *The medication adherence feature comes from [prospective client L], which has 60 clinics, 90k new patients a year.* 3. *I agree it looks dumb but remember the [contact person] is the CEO's kid.*	Good enough for sales, capture business opportunity, funder management
Software-quality focused	Grounded in user requirements, designers' experience or best practices	Users (nurses, patients), designers	High quality software (e.g. Robustness, usability, security)	Focused on operationalization of design requirements and consideration of interdependencies in design	1. *I'd prefer we leave it as-is and turn this into a feature request to make search deeper. ... I don't think the right decision is to remove functionality that mirrors how search works on every other page.* 2. *Version 2.1 has a good amount of stuff in it, and assuming we sign [Client P] we need to carve out a bunch of time for security work. I recommend you prioritize things within the release, and we will attack in that order.* 3. *It would be better to build choice lists with variable image support. I don't like one-offs*	Intuitive design, usability, flexibility, do it right, interdependencies, user satisfaction

Table 4.1 highlights representative quotes for financial-focused and software quality-focused (thereafter, termed quality-focused) attention drivers. For example, financial-focused quotes illustrate how the designers paid attention to clients with big revenue opportunities associated and let the clients dictate the design, even if it contradicted quality. The quality-focused attention driver was grounded in user requirements and designers' knowledge of software quality best practices (e.g. former training, experience, or knowledge from online software design forums). The software quality-focused quotes highlight the designers' preference for software designs with improved functionality. For instance, in the last quote, the iOS Designer's preference was to build the design the right way, even if it would take more time, instead of doing one-offs. Quality-focused attention drivers led designers to invest more resources (i.e. time) in design upfront. To illustrate the tension in attention allocation, I begin with a deeper examination of the two attention drivers.

4.1.1 Financial-Focused Attention Drivers

The financial-focused attention driver was driven by client requirements and investors' expectations for funding milestones. Clients refer to the healthcare facilities that licensed the software from HealthCom. Client intermediaries refer to administrators and managers who participated in the purchase of the software at the client sites, with whom the HealthCom designers maintained contact throughout the licensing agreement. They may or may not be user representatives, depending on whether they used the software. The intermediaries' interest in the HealthCom software provided opportunities for increased revenue for the startup, as they dictated the renewal of contracts with HealthCom. The designers at HealthCom thus felt the pressure to satisfy these clients, particularly if they were introduced by HealthCom's investors.

4.1.1.1 Players and Rules of the Game

I define the players and rules that direct the attention of designers towards financial-focused design decisions as financial-focused attention drivers. As explained above, the designs were created with investors' requests, revenue targets, deadlines set by clients, or marketability in mind. The investors, client intermediaries, and the startup growth and funding rules of the game direct the attention of designers to design alternatives that could improve the financial performance of the firm. For instance, delays in getting a contract signed with the client intermediaries increased financial pressures.

The Customer Liaison also explained how clients' requirements were associated with revenue opportunities, even with existing clients. New requirements were communicated during her weekly client calls, or quarterly client meetings, to renegotiate contracts and were thus tied closely to revenue opportunities:

> Sometimes you [design] because you know the client is going to be able to enroll hundreds more patients on the software more easily. Therefore, the client will pay you more money. ... or 'if we build these things for you, you have to start paying us a little bit more now to fund that front-end design, and you commit that once we build these things, and you're going to go all in and get this thing deployed.'

HealthCom would agree to deliver a feature in the software by the deadline for the client in exchange for promised revenue or contract signing. Features had to be designed as quickly as possible to allow designers to capture the business opportunities. These negotiations with clients hence affected prioritization of designs. The software evolved based on these design requirements: features designed would then be available in the software that was marketed to other prospective clients.

The relationship between the clients and the investors also led HealthCom designers to focus on these players during the design process. For instance, one of the investors began to introduce the CEO to other opportunities outside of the market segments he had been

exploring. She said, "[the HealthCom founders] have this one strategy line... . When I look at [their software], and I see its capabilities... I see [different] kinds of possibilities." In this case, the investor was not focused on the digital business design but on leveraging the software to ensure continued revenue. She saw a potential match between the HealthCom software and the requirements of a employee benefits management company, Client B. The investor believed that HealthCom should leverage its software capabilities to capture new business opportunities instead of trying to only pursue big clients in a certain market segment. She also emphasized that continued revenue was critical to the survival of an early stage entrepreneurial firm. For instance, as one of the investors said:

> I introduced [Client B] to [CEO of HealthCom], they met, and [CEO of HealthCom] was able to build the software... and, go live within three months. That's the speed in which you need to be able to operate. The point of being startup is that you've got to be able to sustain yourself from a revenue perspective. You're supposed to be fast and nimble among other things and deliver the goods.

In response the investor's introduction, the designers had to quickly design a ShareMyCare feature and web dashboard in three months for Client B, which creating time pressures for the designers. The investor expected HealthCom to be able to generate designs quickly to capture these opportunities. The criticality of satisfying this client was intensified because the investor made these client introductions. The designers had to direct their attention to these investor-introduced clients, even if it meant deviating from their software or business plans. The investor explained that startups needed to "make these pivots because they need the capital, and they need the credibility in the client list." The investors and the clients thus became players that directed the attention of the designers through the financial (revenue and funding) opportunities they provided. Designers' attention to financial-focused attention drivers meant designing in ways that enabled HealthCom to capture emerging business opportunities.

The pressures from investors were also felt even though they did not participate directly in discussions within the firm. Frequent updates from the HealthCom co-founders kept them accountable to the investors and sustained attention to the investors and their expectations for the firm. For instance, in emails to investors, the CEO often updated the investors on HealthCom's financing strategies and their financial performance, e.g. "the last 90 days has been very strong for us. We've been focusing on building relationships in pharmacy while leveraging our [software capabilities], and we're finding win after win." In this email, the CEO then went on to discuss the expected revenue and sizes of contracts, number of partnerships, and recent publicity. Conversations with the investors were then shared with the other designers in weekly meetings.

4.1.1.2 Firm Performance

Poor financial performance throughout the duration of this study motivated attention to financial-focused attention drivers. HealthCom experienced immense constraints and pressure to generate revenue. As the Chief Designer announced in an email: "we are still on the track to run out of cash [at Month 26] … [Months 18-21] was exceptionally slow for new business. We fumbled on deals that we anticipated would get us to cash flow positive. … [but] the sales cycle is 9-12 months on average." HealthCom's poor financial performance thus increased the salience of the financial-focused attention drivers and redirected attention making design decisions that could increase revenue. The designers, particularly the co-founders who were directly accountable to investors, were sensitive to business opportunities and even allowed clients to dictate and influence design decisions.

4.1.2 Pressures for Funding in the Early Stages Led to Urgency in Design

The HealthCom co-founders faced pressures from the investors, who felt that they had yet to demonstrate repeated contracts with clients in specific market segments. The HealthCom co-founders had wanted to start the next round of fundraising (Series A) and was working closely with the investors to get the process started to ensure continued funding for firm operations. The investors, however, did not allow HealthCom to seek Series A funding, and instead, suggested that HealthCom seek bridge financing, which is an interim financing option used by companies and other entities to solidify their short-term position until a long-term financing option can be arranged. Platform Designer L explained, "I left was because I didn't feel that the company was going to succeed… The investors did not allow [the CEO] and [Chief Designer] to seek Series A funding because they did not think that HealthCom has demonstrated product market fit." The investors' lack of belief in the readiness of HealthCom to seek further funding during my period in the field further intensified the pressures to capture business opportunities to convince the investors that HealthCom was ready for the next milestone.

The scarcity of clients and long lead times in the contract negotiations also intensified financial pressures, urgency of capturing emerging opportunities, and desire to accommodate the client's requests. At the start of my study, HealthCom had two clients who were also investors (i.e. Clients A and C). During the period of my study (18 months), HealthCom only managed to secure three more clients (Client B, D, E), one of whom terminated the pilot study due to low use of the software. Clients T and P signed the contract shortly after the end of my study. At the meetings, the CEO would share positive news, such as, "Client W is generally excited as an organization, there's a lot of momentum behind that…", but despite the optimism, these contract negotiations took at least nine months to a year before they were finalized, or they would be canceled months into

negotiations or after due diligence evaluation. Clients who did not choose to license the software from HealthCom would end up designing their own versions of the software, as I will elaborate further in Chapter 5. The time taken for clients to sign the contracts increased the challenges of HealthCom in the pursuit of business opportunities. As such, HealthCom co-founders were often attentive to the requirements of prospective clients and would devote resources to accommodating their requests to increase the likelihood of the contracts solidifying to ensure continued funding.

4.1.3 Quality-Focused Attention Drivers

The quality-focused attention drivers directed the attention of the designers to the operationalization of design requirements. I observed several meetings where the designers gathered to review the preliminary design of features. During these meetings, I was surprised by the lack of user-centeredness or evaluation in the design. For instance, in response to another designer's suggestion for improving the ShareMyCare feature given low use, the CTO said, "The feature sells well. Leave it alone". It did not matter to the CTO or CEO that the feature was not used by the patients. Instead, it was a marketable feature in which prospective clients expressed interest. Thus, the design was adequate.

The design discussions often revolved around design attributes. Design attributes discussed could be categorized into functional and non-functional design attributes. Functional design attributes affected the visible behavior of the software. These design attributes, such as user interface, specified what a software should do in terms of inputs, processes, and outputs. Non-functional design attributes referred to non-visible characteristics of the software such as performance, security, and reliability. Given that this study was based in a healthcare setting, legality and security were of concern.

There were a few designers, such as Platform Designer C, Web Designer O, and iOS Designer, who were the most attentive to the quality of the software. For instance, the Platform Designer C was not afraid to question design decisions, particularly when interdependencies were involved. When he first started at HealthCom, he acknowledged apologetically "#newguyproblems" to the group for all his questions on how features were supposed to function, e.g. "It's a bit of a spitball as I'm still trying to wrap my brain around what mechanisms can cause an alert." The alerts feature was complex: nurses could set alerts when patients did not complete tasks or when the patients reported an abnormal clinical result. It was thus tricky to troubleshoot, as erroneous alerts could result from user error or algorithm error. Understanding of the interdependencies between components in the software and how the feature would be used often guided Platform Designer C's design decisions. For instance, the CTO, Platform Designer C, and Web Designer O were in an intense hour-long discussion about the survey feature. They were trying to figure out how best to organize the questions and how to store the data for incomplete surveys. Platform Designer C went on a rant about design decisions of poor quality that were making him anxious:

> I'm concerned that the authoring [user interface] is really broke and that that's going to reveal itself almost immediately when actual patients try to do something with it. My second thing is that the broken [user interface] is forcing us to a conceptual model of surveys that may not be a very good model. … I've never seen things like that ever become anything other than a source of suffering in the end... I've gone to great lengths to make that never a problem, never the thing where you have to go on a giant fishing expedition [to retrieve your data].

Having thought through the negative repercussions of the earlier design decisions, he invested additional effort to make sure that it would not be a problem for the user. This is

an example of the designer directing attention to quality (i.e. reliability and architecture) in design.

Designers paying attention to quality were willing to put in extra work to improve design attributes, such as the user experience, particularly if it inhibited use of the software. For instance, the iOS Designer was providing feedback on the design of the education content page on the iOS mobile application and said to the Chief Designer, "I feel like sliding boxes open as images load is going to be a poor user experience. I would love to keep this simple, but I do think the only way forward is to have the platform return image sizes to us. Otherwise, the entire user interface is going to slide open and jump around like crazy." In this case, the iOS Designer argued that the original design, while simpler, would result in a poor user experience, which was unacceptable to him. He recognized that that redesigning the page require more work, but he felt that the extra work was inevitable to ensure usability of the design for the patient.

4.1.3.1 Ambiguity around Quality and Novelty of Design

Ambiguity around quality led to challenges in making design decisions. The designers did not have healthcare experience, and the clients had limited understanding of the software's capabilities. For example, in the design of a signature upload function, the designers were confused how to proceed due to a lack of specificity in the requirements provided. As the CTO was creating the database and APIs to support the feature, he asked the other designers:

CTO:	What kind of images am I likely to get for this signature business - png and...?
Android Designer:	Whatever you want. something I would like to know is how big you want them
CTO:	WHATEVER YOU WANT!
Android Designer:	Should they have a background?
CTO:	Err

Android Designer:	But yeah, png is a good format
IOS Designer:	I like PNG for this
CTO:	I like PNG..
IOS Designer:	JPEG seems silly
Senior Engineer- O:	PNG seems ideal
CTO:	I feel unqualified to comment on this background thing. I guess transparent images are the most flexible...

As long as the signature was visible, the nurses may not be concerned with the specific file format. This was an example of ambiguity in requirements and quality that arose in discussions. When the CTO initiated the conversation, instead of answering the CTO's question, the Android Designer responded with another question. The Android Designer was tasked to decide the specifications of the signature upload function, but the Android designer, who had no prior interactions with the nurses, was unsure of what exactly the nurse needed. In the end, the decision was made haphazardly based on what made sense to the designers for the software performance (size of file) and personal preference (file type) that was deemed to be adequate for the designers. In a later discussion about this again, the CTO asserted, "I feel like anything we do here is only going to be 70% right at best, so I'm comfortable with whatever seems best to people." There was no one best decision for image format in the CTO's perspective. Thus, what was adequate as a design decision was whatever would be acceptable.

Debates about what constituted designs of quality were catalyzed by varying expectations of the designers. In the above scenario, the designers had a lot of liberty to decide how the designs should be operationalized. These design conversations helped the designers proactively identify possible defects, security flaws, and quality compromises. A pursuit of quality designs was associated with spending more time in discussion and design. It also meant thinking about the implications for design in the long-run.

Another example with more severe repercussions was a discussion regarding the protection of patient confidentiality in design. HealthCom would be liable for any

confidentiality compromises. However, this was difficult as there were no objective benchmarks for patient confidentiality in design. The following conversation about the ShareMyCare feature highlighted how the iOS Designer was focused on the quality (legal implications) of the design, whereas the Chief Designer was focused on what was adequate for the client for now. In addition to the Web Dashboard, Client B had requested for a ShareMyCare feature, so that their patients could share their care plans with their family or friends. The iOS Designer was uncomfortable with the existing modeling of patient relationships in the software, as he felt that it had privacy and legal implications:

> iOS Designer: The more I think about it the more it seems like "ShareMyCare" needs to be "divorced" from “Relationships”. We tried to shoehorn two features into one. And they are nuanced enough that there are some strange edge cases occurring because of it
> Chief Designer: I think you have fair points, but I don't think the edge case is worth a rewrite at this moment.

Although the Chief Designer agreed with the legal implications, he questioned the need to change the design. He did not agree with the probability of the hypothetical scenarios proposed by the iOS Designer. More importantly, the existing design met the design requirements provided by Client B, which was the immediate priority. The iOS Designer continued to provide examples to justify the need for redesign. The Chief Designer told him that the existing relationship was set up only so that the user could get through the enrollment flow, i.e., creating a user account, which was a growth metric tracked by the client and the investors.

The Chief Designer argued that a simplified representation of the relationships would suffice for Client B’s use and asserted that the redesign would lead to them missing deadline. The Chief Designer’s attention was thus on the financial implications of redesigning the permissions model. To support his point, he referred to the Android

Designer's design as adequate. The Chief Designer did not see any value in making the improvement to the design.

> Chief Designer: But we're not rewriting at this junction, that's my point. The way Android Designer implemented it in the Android app is perfect. It exposes the feature, without violating privacy and it throws a simple popup if they don't have permission. The business case for [Client B] is solved as-is and the "share my care" feature is unique to our software. If we need to better support the idea of relationship down the road, we can. [Client B] won't be jeopardized and we still have our share care feature intact
>
> iOS Designer: I disagree. these are two different features and they don't fit well together
>
> Chief Designer: And you're proposing what exactly? that we miss our [Client B] 400 employee roll out and 8000 employee roll out Oct. 15th, to rewrite a problem that doesn't really exist at this junction?

Additionally, he believed that this was a feature that only HealthCom had, There was no comparable design in the market upon which quality could be judged. It was also difficult to determine if the relationships were modeled effectively, as relationships in real life were complex. He thus tried to direct the iOS Designer's attention towards design requirements critical for meeting the client's deadline, and the financial implications of design choices.

4.2 Tension Between Financial and Quality-Focused Attention Drivers

Every designer in HealthCom was exposed to both attention drivers. The designers who did not interact with the investors were made aware of the firm's poor financial performance and the need to increase revenue. Furthermore, the relationships between HealthCom, its investors and clients were complicated because some clients were also investors. For instance, the CEO of Client C, who was also the first investor of HealthCom said, "As a client, I would rate HealthCom a A-, but as an investor, B+, as I have yet to see my returns on investment." The quote highlights the challenges that could result with individual taking dual roles. The same individual had different expectations for HealthCom

depending on the rules of the game against which he was evaluating HealthCom. As a client, he was looking for software that would improve his patients' health and satisfaction. A higher quality software that met the users' requirements would better achieve this objective. However, as an investor, his attention was on his returns on investment and the ability of HealthCom to demonstrate sales quickly. Sales would be achieved through designing features quickly to capture emerging business opportunities. Given the limited attention and resources of the firm, the tension between quality and financial-focused design alternatives created disagreements amongst the designers.

To integrate the attention of the designers, the co-founders set up communication channels to ensure that the designers were kept updated on the deadlines and revenue targets. For instance, the CEO would conclude the weekly meetings with a segment called "shaking the money tree," during which he would provide updates on the status of his contract negotiations and anticipated deadlines for designs. Understanding of the relationship between design and revenue was important for making sure that designers were "on the same page" about design deadlines and requirements. The Customer Liaison expressed her frustration with the iOS and Android Designers who were worked remotely and sometimes could not understand the rationales for design:

> [The iOS and Android Designers] didn't feel like being part of team. On a daily basis, it made it difficult to get things done. The reason is that they didn't see the marketing and the client activities. Like I'd come out from a call with a prospective client and complain about their requests. And I'm pulling people in to make things happen and they didn't get to see that. So, they will ask things like, "why do I have to do things this way?"

The designers believed that if they created designs for the prospective clients' requirements, they could increase the likelihood of a successful contract. As such, the designers were often asked to create features quickly to motivate prospective clients to sign the contract. Absence of the iOS and Android Designers from this communication resulted

in lower salience of the financial-focused attention drivers in directing their attention towards financial-focused designs.

Attention to financial-focused opportunities often resulted in quality compromises. In response to an opportunity with Client B, HealthCom had to design a web portal for patients' use. The designers had assumed that the clients of HealthCom would be large enterprises that would require integration of the HealthCom software into the clients' existing health IT software. As a result, they had not intended to design web portals for patients or nurses to use. Because of Client B, the designers quickly put together a web portal for patients to close the deal. The Chief Designer explained, "The client conveyed it [as a deal breaker] ... We really wanted [the client] We made it pretty quick... That continually haunts us ... the [web] dashboard was never really intended to be patient facing". To capture the revenue, they had to reallocate resources within the firm to design the web portal within the deadline. To design the web portal quickly, they took an existing web dashboard that was meant for nurses and created workarounds to accommodate the needs of users.

Deference to the clients' requests was critical, in HealthCom's opinion, to getting the deal signed. Just as the above example showed how a client's request can lead to reallocation of attention within the firm, the interactions with the client intermediaries affected how designers chose to design. The desire to satisfy the client could be at the expense of quality. For instance, in the following conversation, even though the conversations offline demonstrated how the designers vehemently disagreed with and mocked the design of the user interface proposed by the intermediary, they proceeded as directed by the intermediary because they did not want to defy him:

CEO: Also, this is [Client A's] logo and they were pretty set on having [a particular tagline] showing. If you want to deviate from that, you'll have to explain to them why.

Chief Designer: Don't really feel like "coaching" them at this point. It just looks pretty amateur. The logo looks nice, but the tagline looks slapped on there in Arial [font].

[The co-founders continue to criticize the design]

CEO: I agree it looks dumb but remember - [the intermediary] is [Client A's] CEO's kid

The conversation highlights how the designers were so focused on the contract with the client that they did not want to risk upsetting the intermediary, particularly because he was the son of Client A's CEO and proceeded with a design that looked "dumb."

The CEO prioritized designs associated with revenue increases and the possible entry to new market segments. In the pursuit of financial-focused designs, the designers' attention was often on functional design attributes, and specifically, creating more features to meet clients' requirements. As the CEO said, features such as these that were "important from the executive perspective" were pushed to the top of the priority list. Because of the opportunity that arose, they had to move the web portal to the top of the priority list. The need to capture the client's business meant rethinking how to design a dashboard quickly and re-examining their product vision. The Client B opportunity came to the firm after Clients A and C, when the HealthCom designers still had a tentative product vision in mind. However, Client B was introduced by the investor, who had advised them to adapt their software quickly to meet the needs of their user. These financial-focused design alternatives directed the designers' attention to functional design attributes, such as scope of function, the user interface, and to a less degree, user experience.

Further examples illustrated the tension between making design decisions that were financial or quality-focused. In the following example, the iOS Designer asked the CTO

whether the existing API design should support icons for recording patients' moods or pain level. The norm in healthcare was to provide a scale with smiley faces that patients could press to indicate their pain level or moods. There was a disagreement about whether they should support this client request in the upcoming version of the design, given the deadline. The Chief Designer argued that the icons were not necessary for version 1.0, as they could meet the client requirements without having to put in effort to support icons:

> iOS Designer: [To the Chief Designer] this came up during the [Client C] meeting I was at. Specifically recording pain levels and happy / smiley faces. Or mood. If we need to support either "pain level" or "mood" tasks for version 1, we will need to show images for each choice.
>
> Chief Designer: I heard that too, but I don't think it's a high priority for version 1.0. What if you just did it on the [mobile application] side and assumed that mood is one of those universal tasks that gets its own modal kind of like blood pressure or weight?

The iOS Designer had thought that it was a nurse who requested the icons for version 1.0, as he was simply responding to the request he heard at the meeting with Client C. He was thus thinking about fulfilling the user requirement to improve the software functionality. Even though the Chief Designer heard the requirement, he interpreted it as non-urgent for this version. His focus was on meeting the deadline, so he suggested a workaround the iOS Designer could adopt. The iOS Designer resisted because he felt that it had repercussions for the future designs:

> iOS Designer: It would be better to build choice lists with variable image support. I don't like one-offs
>
> Chief Designer: My concern is the extra engineering effort to add/pass image icons from the platform side (for version 1.0). Both for CTO and [our contractor]. I do think it will be a nice feature though and should take priority in the version 1.1 release
>
> iOS Designer: We have to be careful about minor point releases "breaking" the JSON returned by the server

CTO: [proposes a workaround solution] It would be nice, but we're really at the point where holding the line of scope is a Good Thing (to me)

Chief Designer: We have 10 days until the date we're targeting to launch and be prepared for [the major health IT conference]. I feel like this is a version 1.1 feature and you all are spinning cycles on stuff where there are more important things that need attention. My opinion is, if the app is done and tested before the 19th, then I think you should add it in because it will help usability and presenting better. I think it's a good feature, just very cautious about adding more overhead until we're rock solid for the pilot.

CTO: I'm fine with doing nothing with this for now unless we think doing something saves us time today. [The Chief Designer] raises good points.

Notice the difference in what the iOS Designer, CTO, and Chief Designer deemed to be important and needing attention. The iOS Designer argued that he did not like one-offs that would affect quality. The CTO and Chief Designer were focused on the deadline for the upcoming Health IT conference during which they would market their software and acquire clients. It became a question of investing attention and resources to designing the feature properly and immediately, or to postpone the investment of resources, to meet the client's deadline. The iOS designer continued his argument:

iOS Designer: If it doesn't take much time, I'd prefer we get the JSON format for this stuff setup before launch. Otherwise we will need to version the API or something for v1.1 But that's all up to [the CTO]

CTO: Well, I'm not sure that's necessarily true - I assume that adding elements to json will not break the iOS app. If it does, that's probably a bigger concern.

Chief Designer: If you're just adding additional data to the JSON response, how does it break the app? I thought it just ignores the stuff it doesn't use.

iOS Designer: Its changing a string to a dictionary. I could parse this one structure assuming it could change in the future. But I can't do that to every structure the API returns. It's one thing to add a new key/value pair to a dictionary. It's another to change something's type completely

CTO: Fair enough

Chief Designer: Yeah, I understand

CTO: It will take me 15 minutes to do something with that, although, I must say that...uh

Chief Designer: You guys have my opinion on it so yeah. I think it will improve user experience, but I think we need to be really mindful of the next 10 days, that's all.

iOS Designer: I'm happy to do whatever. I just want us all to be aware of how disruptive changing the server's response format can be if it is a bigger change. Especially when users don't have to update their apps and one server has to support all versions of the app.

While the iOS Designer was willing to concede, he wanted to make sure that the designers were aware of the repercussions. The Chief Designer tried to redirect the iOS Designer's attention to designs that were needed for the upcoming deadline. The CTO, similarly, preferred not to invest the resources, unless needed. He was willing to allocate attention to investigating the problem after being educated on the repercussions by the iOS Designer. It was ironic that the entire conversation lasted more than fifteen minutes when the CTO stated that the code would only take fifteen minutes to fix.

The tension between financial and quality-focused attention drivers further drove tensions in design. Three key categories of tensions emerged. Creating designs to capture business opportunities quickly meant making design decisions that compromised the quality of the software. Designers in HealthCom who focused on quality were viewed by the co-founders and CTO as over-optimizing quality unnecessarily. The cumulative impact of these decisions may also inhibit financial performance or future changes to the design. The tensions hence created disagreements within the designers between the value and cost of the extra work necessary to "do it right" versus the value and cost of creating designs quickly. Given that HealthCom was focusing on increasing revenue and contracts, the limited attention and resources of the designers meant that they had to choose to focus their attention on either financial or quality-focused designs.

4.2.1 Compromising Quality to Meet Immediate Deliverable

Attention to financial-focused designs, amidst time pressures and ambiguity around quality, meant opportunities for compromising quality in the design. One of the most common themes in the discussion was “adequate” design. As the CTO explained, as a startup (i.e. rules of the game for startups):

> You're willing to ship with things and the lack of things that no large company would ever be able to tolerate. Basically, you cut corners. The corner cutting hopefully is larger on features, but it can also be on things like quality. A startup might say: "You know what? If the app just explodes for 5% of people, then maybe we don't care about that..." Whereas a more mature company might think that was ... Even at a startup probably ultimately will think that's not acceptable, but in the initial stages you might make all kinds of compromise.

As the CTO explained, compromises were made with respect to the scope of the feature and quality. As a startup, the rule is to deliver a software quickly and seek feedback and iterate as needed. Although client requirements typically informed the design of user interface or user experience, the clients typically did not provide the specifics of how the design is operationalized on the backend. As the CTO justified the decision to limit the scope of the design and constrain disease states and care plan to a 1:1 relationship in his design of care plans, he said, *“[it] may turn out to be an overall weakness of the design. There are some things here we don't know*.” Although the designers could clarify the requirements with the clients, they did not, as they believed that more attention and resources allocated to improving the quality of designs would not necessarily correspond to higher value to the firm in the immediate future.

Instead, the designers made design decisions that was just adequate for meeting the client’s requirements and for delivery by the client-imposed deadline. Any improvements or changes could be postponed until the client requested for it or when they had more resources. As exemplified in the arguments in the ShareMyCare feature discussions, the

Chief Designer said, “If we need to better support the idea of relationships down the road, we can.” These statements indicated the lack of desire to improve quality of the designs now, despite knowledge of the shortcomings of the design. Doing so allowed the designers to capture the business opportunity, while deferring investment in design work. Thus, the disagreements often revolved around whether a design investment that would go towards designing a more robust software could be postponed. As the Chief Designer said in response to a proposed improvement, “If [the client] needs more, we can give it more thought for second release. but I still think this is sufficient for [the client].” The CEO has also made similar decisions, “I think the user experience is OK for version 1.0. This isn't a super important part of the app.” The focus was on meeting the mandatory requirements from the client led to the postponement of investments in other designs until the firm had more resources or when requested by the client. As the Android Designer concluded, “I really think the time constraint is much more the problem than the money or expertise constraint.” In HealthCom, the designers with contact to the clients and investors often dictated the design and the deadlines.

Given the need for high reliability in healthcare, it was revelatory to see the lack of focus on reliability, security, and other quality of the software. Instead, it was acceptable to the clients and investors to have lower quality software. Design work was often rushed before a deadline for existing or prospective clients. When things could not be fixed, workarounds were always created to cover up defects. These workarounds were adequate for the CEO’s presentations to the financial-focused players. For instance, the CTO informed the CEO that when he updated the code on the server, the patient account creation function stopped working as intended. The CEO’s response was not one of concern for the existing clients, but instead, *“*I do have another marketing presentation at 1pm. I can hack together as I did this morning.” Even though he would be presenting a defective software,

it was easily resolved by reassuring the prospective clients that his designers were working on a solution. As the CEO smiled and said, “Luckily I'm pretty good ah hand waving.” Time constraints led to accepting quality compromises as long as the designs were still adequate for marketing purposes.

The CEO and Chief Designer’s attention to financial-focused designs led to the focus on deadlines instead of quality. For instance, for the first version of the Android mobile application, the iOS Designer was tasked to design the Android application to meet the client’s deadline. He had limited experience in designing Android applications. Thereafter, the pressure was on the iOS Designer to complete the Android application quickly, “Since the press release is going out today and if you feel ready, then we should submit the Android app. If we find other big bugs, no big deal. we're not going to have any patients using it on day 1. It’s really just optics.” The Chief Designer’s comment provided an initial glimpse of how client-imposed deadlines shaped attention allocation in design. A to the Chief Designer, it did not matter that the Android application was not functioning up to standard, as it was simply “optics” (i.e. for publicity). As the Chief Designer anticipated, no patient would be using it on the first day. At this point in the study, HealthCom had only licensed the software to two clients-investors. Designing for optics was representative of the desire to satisfy the investors and the clients of their ability to deliver on the product in a timely manner.

Similarly, questions about non-functional design attributes, such as the logic behind algorithms, became evident when the designers started designing based on the requirements. The designers often worked without specifics of how the feature would be operationalized. For example, the designers did not know what sorts of surveys would be administered and how complex the algorithms had to be. As the Chief Designer said about a design decision, “I feel like in the short term it’s not a major problem. And the only

implication is on the mobile side. To this many organizations, they may have a degraded experience cover until we solve it." In this example, the Chief Designer was willing to have a lower quality software to ensure that they met deadlines. The Chief Designer felt that the compromise was acceptable for the design product to be submitted to the client and would affect only users of the mobile application. As such, his decision was to postpone investment in this refining the design until they had the resources to build it robustly.

The alerts feature was another example of a feature that had quality compromises. The CTO explained that the design was just not "thought through all that cleanly and at times it just gives people the appearance that the whole thing is just not working that cleanly... Which is really, fundamentally a product problem because you designed something that did not work well." The lack of attention to the interdependencies and how the different components of the web dashboard should work with the alerts algorithm meant that the feature often did not work well. Alerts were being activated for things that did not warrant attention by the nurse. The erroneous alerts were interpreted by nurses as software defects. What was needed to rectify the design was an understanding of the nurses workflows and use of the feature. Unfortunately, the financial-focused attention drivers directed designers towards the design of the alerts feature, but not necessarily improving the feature to make sure it worked for the users.

Given the limited attention and time of the designers, the cofounders and CTO constantly questioned the likelihood of the negative repercussions materializing. There were multiple instances in which the designers wrestled with the cost and benefits of selecting a workaround solution versus a quality design. For example, in the survey discussion, Platform Designer C went on to say:

> That makes me anxious, but there are things about this particular case that I think vindicate my anxiety a little bit... I don't think those are the kind of things patients are going to encounter all the time. I think most of these things are going to be really simple.

In the end, the designers chose to not worry about the hypothetical cases they initially brainstormed, as they did not believe that they would happen often enough to affect the software quality or to warrant further attention.

The CTO sometimes deliberated between costs of investing in the design now versus redesigning the software in the future. While he advocated for the most minimal approach, he understood the tradeoffs with respect to quality.

> Do we make [the design investment] now or make those changes in 4 weeks? I think that the degree to which we have conflict is often around, you know, when do we tackle the direction we are going to make? Do we have a contract? Or an advance [payment]? Do we stop what we are doing on this release and do something different? It is less about what to do but when to do it.

Time constraints resulting from the client-imposed deadlines often led to disagreements about whether investment in design work was warranted for the client's requirement. If they made design decisions that compromised the design quality now, it would inhibit the scalability and growth of the software in the long-run. Yet, if they did not make this tradeoff, they could invest resources in the design unnecessarily and fail to meet the client-imposed deadline. The decision was sometimes a difficult decision; taking the time to design the software correctly in the present before the software became too complex might be the efficient approach in the long-run, but they did not have the resources to do so. Yet, he must look at the short-term and decide if this feature was adding value. If it could be postponed and the resources could be directed to another design that yielded revenue returns, that would be considered attention and resources well invested. In a response to the Chief Designer about his request to get patient counts for the clinic, he said:

> [Getting patient counts] is not going to be something that we can do at the moment. I can get into why that is what you want, but the bottom line won't change. I think at some point down the road we can do that. The issue is that while we can do that in a reasonable amount of time right now, once we get bigger it will start to either be brittle or a time consuming in a system time sense. I think when we do our next iteration... that will be easier to get at, but for now hopefully you can live without those #s.

The conflict for the CTO was whether to invest the resources of his designers towards improving a design now or to direct their attention towards other features.

4.2.2 Over-Optimizing Software for Quality Unnecessarily

Designers' attention to quality and non-functional design attributes, such as scalability of software, was often viewed as over- optimization. Designers in key decision-making positions (e.g. CEO, Chief Designer) often viewed this attention as unnecessary. The tension was partly driven by the optimism of the CEO, who often reported at weekly meetings about upcoming contracts that could potentially "drive a couple million dollars". However, to earn a couple million dollars, the software would have to be able to support a much larger user base than the current size. This would require the designers to focus on the non-functional attributes of design. The uncertainty about when such a need would materialize made it challenging for designers to allocate attention and resources towards building a quality, scalable software. As the Platform Designer L described:

> At HealthCom, we really have super small sets of data. We think at scale, but if you don't see the patients [using the software], it is hard to design. You want to design for scalability, but how optimized do you have to be, if you only have 100 users using your software? It really requires a change in mental model – you want to see 10,000 users, but really everything is a pilot. And so, you have to act like you are designing for many pilots.

The Platform Designer L's interpretation of the design process in the early stage of the startup was that designing features to support the capturing of multiple business opportunities was akin to designing for multiple pilots.

Despite the known defects in the software, attention to quality was still viewed as over-optimizing that was a waste of time. For instance, when the iOS Designer recommended simple changes to the user interface to improve navigation of the app, the Chief Designer interrupted, “You’re optimizing for speed, which is not the right decision for this app.... You’re overthinking this a lot. This is not an app that’s going to be always on or with a high volume of activity.” While the HealthCom software may not have had many users at this point, they had intended to grow to tens of thousands of users in the near future. As such, the iOS designer recognized that the current design approach would not be suitable for growth. Though these opportunities were associated with tens of thousands of users, they never materialized in the duration of this study.

Similarly, in the example of the ShareMyCare permissions design in Section 4.4.1, the debate between the Chief Designer and the iOS designer occurred because the Chief Designer felt that the iOS Designer was over-optimizing the design:

> Chief Designer: You are complicating a feature that should be simple
> iOS Designer: No, it's not, it's a privacy issue.
> Chief Designer: SIMPLE. The feature as it stands is needlessly complicated

The Chief Designer did not believe that the extra design resources and time invested to ensure that the design of the permissions model on the ShareMyCare feature to mirror reality was worthwhile, as he did not see the threats to the firm. Hence, he felt that the iOS Designer was over-optimizing by directing too much of his attention to the quality. The iOS Designer felt this was not a status update on a social media application, but instead, a healthcare use in which there could be negative repercussions for the patient to poor design. Secondly, he believed that the designers would not revisit this issue and correct it in the future; to his point, he stated in a separate interview that the designers often deferred design investments and never came back to improve the designs unless they were marked as a

defect or associated with revenue increases. As such, he called the designers out on this promise. The CTO did not agree with the significance of the problem but proposed a workaround solution to distinguish between relationships in ShareMyCare. Undeterred, the Chief Designer continued:

Chief Designer: Look, everything I saw in Android Designer's android app... the implementation in the Dashboard... and the goals for [Client B] – they are all satisfactory for a 1.0 release. There is some polish, but I think you're creating a lot more issues than are there. I'm not saying that we should ignore them... I think we can thoughtfully improve the feature over time.

iOS Designer: If we don't fix these now, I highly doubt we are going to fix them later.

[The CTO and iOS Designer continued to debate about what the problem is and what needed to be done to resolve the problem]

iOS Designer: Eh, I disagree that the current implementation is satisfactory for version 1.0.

Chief Designer: To assume we're never going to fix things is a short-term concern and a reflection of resources. You interviewed another backend engineer this week!

The designers frequently disagreed on their interpretations of the criticality of the design changes. Note that throughout the problem, the Chief Designer's attention was directed by the financial-focused attention drivers. He did not try to understand the problem the iOS Designer was alluding to. Instead, he iterated multiple times that the changes were: (1) not urgent, (2) not a real problem, and (3) reasonably fixable with more resources in the future. In the end, the CTO concluded that changes would not be made:

CTO: I think that making changes on the backend are not realistic at this juncture, but I also don't think anything iOS Designer proposes from the mobile UI perspective requires that.

Chief Designer: I just ate lunch and you're giving me indigestion.

iOS Designer: GOOD. Aww, I thought he stormed off

Chief Designer: I DON'T RAGE QUIT!

The above concluded the heated debate verbatim from the chat transcript. It highlighted the difference in the designers' attention allocations: the CTO focused on what needed to be done to operationalize the design with minimal effort. He concurred with the Chief Designer that the problem raised was not a big, show-stopping one, and was fixable in the long-run. The Chief Designer repeatedly used the argument of immediate value to convince the iOS Designer that such an investment would be unwarranted. The iOS Designer, on the other hand, was focused on the interdependencies in the design and the bigger implications for software quality.

The use of hypothetical use cases in the discussion often drove these disagreements. For example, in another disagreement about the relationships model on the ShareMyCare feature, the iOS Designer was advocating for a closer examination of the current design. Specifically, if patients and their spouses got divorced and no longer wanted to let their spouses access their care information, how long would the software take to update and reflect that change in permissions? The iOS Designer was arguing that fifteen minutes was too long and should instead be real-time due to security issues. The CTO responded:

> So, HIPAA [the Health Information Portability and Accountability Act] is a slippery beast; but if that change effectuates in a reasonable period of time, then there's no problem there...Implementing this so that it is immediate is actually pretty easy... We pay the price by having to go look that up on every request. And we couldn't cache that because, well, because of [this problem].

It was difficult to determine if the iOS Designer was over-optimizing due to the ambiguity in quality, due to the lack of objective guidance in the HIPAA law. The CTO was conflicted between making the permissions update real time and the possible software performance issues that could result from this update:

iOS Designer: I worry about security issues and the situation above where we have two devices open and my mom changes access and then I can't see that change for 15 minutes. To a patient, that will be a bug.

CTO: [CTO suggests a workaround] That personally doesn't concern me, but I don't feel terribly strongly about it. It's a sort of unusual case, in that even noticing it requires a fairly concerted effort. But, if the consensus is we need to change it, it won't be hard to do.

The iOS Designer was arguing that the longer time between updates would lead to security concerns. The CTO's question was, if the fifteen minutes would be a reasonable enough time period for the law? The difficulty with these decisions faced while operationalizing the design was that oftentimes, there was no clear benchmark for what constituted secure or of high quality. Additionally, while it would be easy to fix for an immediate refresh of the software, it had implications for technical performance of the software, as the server would have to constantly check for updates and would thus compromise software performance in other ways. After some offline discussion between CTO and Designer P, the CTO agreed to make the change: "It only takes a couple of lines of code to do the check, and it's in a very fast part of the software. It happens to be that we can just look up fresh the thing we derive from the token. So, the underlying authentication code stays precisely the same." The CTO realized that making the change was easy and would also concurrently resolve other design issues they have with other features. Upon further investigation, the iOS Designer realized that the update would occur only after six hours, instead of the fifteen minute he thought initially, which made it a bigger concern. There could have been a security breach that has now been averted because of the persistence of the iOS Designer in the discussion. The additional work associated with reducing the token refresh time was viewed by the CTO as redundant. As usual, the debate was about the value of the additional effort for design versus the probability of the event happening.

4.2.2.1 Changing Procedural and Communication Channels to Reduce Attention to Quality

The HealthCom CTO and Chief Designer had initially asked designers to work together to define the scope of requirements for features. The goal was to help designers understand the rationale for designs. It was hoped that this collaboration would direct the attention of the other designers (with no client interaction) towards understanding the urgency of financial-focused designs. These features and ideas were initially discussed during weekly meetings with all eight designers.

However, the influx of client requests resulted in these meetings being characterized as too "distracting" and "time consuming" with "too many cooks in the kitchen." Decision-making was then shifted to a four-person prioritization committee, comprising the Chief Designer, CTO, Customer liaison, and occasionally the CEO. Together, the committee members would decide what features should be prioritized for meeting client requirements and how these requirements should be designed.

As a result, the designers' input in design discussions was reduced. The CEO and Chief Designer saw little value in the designers' input towards designing the features to meet revenue targets. Instead, they wanted to direct the designers' attention towards designs that were adequate for the client. While the individual designers still had some flexibility over how they designed components of the software, they did not have a say in what features were designed. Some designers, such as the iOS Designer, were disappointed that they were excluded from the discussions that shaped the design. Although they recognized the need for efficiency, they feared that this would result in an inferior software. Desire to reduce discussions about optimizing the software quality led to reduction in

designers' involvement in the design discussions, and the movement towards a more top-down design process.

4.2.3 Inhibiting Future Growth of the Business and Software

The earlier focus on financial-focused designs meant compromising software quality to deliver the designs within the deadline. The unsustainability of focusing on financial-focused designs over quality-focused designs also became evident, as poor quality would inhibit the ability of the software to serve as an enabler of funding over time. In the following conversation, the CEO was updating the Chief Designer about his contract negotiation progress with clients P, T, E and H. In response, the Chief Designer said:

> Chief Designer: I am feeling anxious. We have trouble [getting users to use our application] one month after launch - only 8% are using the application. That is bad engagement. No number of buttons I'm designing will make people click.
>
> CEO: I don't think you are wrong. But solving this should not be a problem. My observation of Client B – with Customer Liaison being more involved in marketing… I think we can run campaigns to increase use of the application. [He goes on to propose other possible solutions] We spend a whole release: testing / organization to increase confidence that it will be better.

Here, unlike in the previous section, the poor software performance was becoming salient to the Chief Designer. He could see how the inferior quality was going to affect their ability to reach revenue targets, if not addressed in a timely manner. They were experiencing poor use levels with Client D and Client B. His attention was thus directed towards software quality. Conversely, the CEO was arguing that the resources should be directed towards financial-focused designs, such as developing interoperability capability for prospective client H. The CEO felt that with the new clients, there would be gains in financial resources, which would allow them to direct attention towards these software performance issues in

the future. By then, with the influx of resources, solving the quality issues would not be a problem. He felt that there were manual workarounds that would not take time away from financial-focused designs.

The co-founders differed on their perception of the urgency of the deficient performance, however. The Chief Designer asserted that if the quality issues were not resolved, the design of other features, such as data analytic capabilities, or ability of the software to handle hundreds of thousands of users, would be a waste of effort, as no one was using the application.

> Chief Designer: Throwing care managers at it is not a solution. It is not reflective of long term use case. I'd be interested to see how they settle patient invites [to get them enrolled on the software].
>
> CEO: Sure. All I'm saying is that it's a process and is simple. We just have to bring people in - we do as much as we can ... we know engagement level is not 50%. [Client B] is aware of actual engagement rates. So, what should we do? … My fear is more about reliability. I think it's worth probing to see how easy it is to crash with so many patients on. My concern is that this is not robust. We broke Client B- we need to do better. From what I've observed, getting data out of this is important.

The CEO continued to argue for manual workarounds to the problem they faced. He separated the problem of poor use from bad design and did not believe that the bad decision would be inhibit marketability of the software. In fact, he saw attention allocated to addressing these concerns as a distraction from making progress on financial targets. He needed the designers to design the new features instead of improving the old:

> Chief Designer: You are not listening to my experience. The client's data sucks. How do you prevent a Client B [failure]?
>
> CEO: So do we take ownership marketing plan? What I am saying is we have 60-90 days to do whatever we want. We are smart people we can fix it. Here is the hot deal. We need to do what we can - we are doing what we can to generate enrollment for a week then deal with the integration of software.

The CEO once again downplayed the criticality of improving the software quality. He was optimistic that with manpower and financial resources, they could resolve the issue, for instance, through marketing and education. In response to the Chief Designer's question about how to prevent a similar failure, he was unable to articulate a solution because he never really understood the problem.

The co-founders typically paid more attention to the financial-focused designs, but here, the Chief Designer recognized the problems of continued allocation of attention and resources on financial-focused designs. Conversely, the CEO had moved on to what was more important in his perspective, i.e. new features supporting data analytics:

Chief Designer: Well that's not going to help me if at the product meeting: prioritizing everything to do with integration? this deal influences how we are prioritizing this. I'm communicating to you the resource needs: sometimes it's all about the customers. but let's also be reflective about Client B. Customer Liaison is marketing it but I don't think there is a silver bullet

CEO: Sure. Let's be smart this time. Let's do everything we need to do — be smarter about our mechanism.

Chief Designer: No. With Client H, you came in and started telling us what we need to do. Stop letting sales dictate our resource allocation. You sales guys do not have any idea of what we are building and, yet you come back to the product meeting and start changing our priorities?

CEO: I understand. We have [a large deal] coming up. I would like to think that we are smart enough to solve this.

The above conversation illustrated differences between the co-founders in attention to poor software performance. These two co-founders should be expected to be in greater alignment in attention allocation than the other designers, given their interactions with the investors. Yet, they disagreed about how attention and resources should be allocated among design alternatives. The CEO's recommendation was "to be smart this time," which

provided little guidance on how to resolve the problem with the software quality. The CEO, who was no longer directly involved in the design process, instead looked to the next design request dictated by the prospective client.

As this chapter has shown, the financial-focused attention drivers can direct attention of the designers away from quality-focused designs alternatives. However, continued attention to financial-focused designs was not sustainable. The need to design the software with increased attention to quality resurfaced when HealthCom eventually managed to sign a few major clients. The CEO announced, "Our value to them will be based on both our ability to enroll those people and demonstrate our value by demonstrating stickiness and engagement. So, we're really going to be putting the software to the road." As the earlier conversation highlighted, the use of the software was low. In the design of the features, the users were often absent from the discussion. Thus, it was difficult for the HealthCom designers to know what and how to design for patients' use. As such, the lack of attention to software quality-focused designs resulted in a poor performing software. The signing of contracts with the major clients meant that the software now had to catch up with the business models, i.e. the design had to be able to serve the thousands of users to demonstrate value.

Anything that the CEO deemed to be inhibiting the negotiation of contracts with clients would be prioritized for design. For instance, the CEO and Chief Designer attributed the failure in Client D's use of the software to poor content created by the client. In the following conversation, he and the Chief Designer disagreed about how to improve the content quality:

> Chief Designer: Content should drive experience. I didn't expect us to get into content business, but I guess we have to. If we are going to be on the hook, we might as well do the whole thing [i.e. software and content].

CEO: So, I think we are getting into the content business whether we like it or not. At least through [working with] partners. The doctor is doing a good job, isn't he?

Chief Designer: Well…… What' s the content we are doing right now? What's the goal? Is there a client driving all this?

Here, the Chief Designer was reluctant to spend money on developing content, even though he recognized that the poor content Client D used was one of the key factors for the low use of the software by Client D. If the education content provided on the software was not engaging or informative, the nurses would not be likely to share it with the patients, and the patients would not read it. In response, the CEO hired a doctor to develop personalized content in diabetes for HealthCom's marketing purposes because the CEO and his business consultant believed that chronic disease management (specifically diabetes management) was a market to target. The CEO believed that the lack of clinically validated content in the software hindered his ability to sell it.

Without really understanding if the lack of clinical content was a deal breaker, he proceeded to invest in the doctor's time. His justification was that: "not having content delays revenue. Makes it harder for someone to say yes [to a deal]" and thus his solution was to "go find some content partners, license that s**t…We will pass [the cost of the content] through as a software fee." The Chief Designer recognized that investment of resources in content design would not generate immediate value if it had not been requested by a specific client. For instance, the CEO spent money on diabetes education content, but if the client were a surgery center, it would require investment in another set of content.

The disagreement here showed how even though both co-founders were focused on allocating resources to financial-focused designs, they disagreed on what feature should be prioritized. While the Chief Designer agreed with the problem, his proposed solutions were different. The Chief Designer continued, "I struggle with this decision [to invest resources in licensing the clinical content]. Where you are going makes a lot of sense, but until we

remap our product, there is no point licensing the content." He proposes an alternative design feature, called alerts, that "could provide value to begin with, without licensing." He felt that they could digitize and improve the client's current content without paying for it, by referencing information available online, without paying for it. Instead, the designers should direct their attention and resources to designing better algorithms and smarter features. The Chief Designer believed having more features, such as alerts, could improve marketability of the HealthCom software.

The CEO was not giving up. He suggested postponing the decision. Later in the day, the CEO received positive feedback from a major client interested in licensing the HealthCom software. He found out that the client's "business model [did] not [require us to have content]. So, [they did not] have to worry about content anymore." Notice how quickly the CEO changed his mind. In a couple of hours, a major client validated the value of the HealthCom software even in the absence of content. The conversation exemplified how the CEO's attention to design shifted according to the cues he was receiving from the players.

4.3 SUMMARY

In this chapter, I introduced the key attention drivers and the tension between these two key attention drivers. I highlighted how the tensions between financial and quality-focused attention drivers led to design disagreements about design decisions that compromised the quality of the design, over-optimized quality unnecessarily, or inhibited growth of the firm. The financial-focused and quality-focused designs did not appear to be complementary in the short-run due to the deadlines imposed by the clients Ambiguity in quality also offered designers the opportunities to take shortcuts in designs to conserve their effort. In the next chapter, I focus on the design actions and impact on design product.

Chapter 5: From Design Action to Design Product

In the previous chapter, how the attention of the designers was directed to the various design alternatives. In this chapter, I focus on the design actions and design product. Overall, the following design actions were because of the financial and time pressures. These design actions included: (1) reframing, (2) simplifying and (3) coupling, all with the effect of (4) postponing investment of attention and resources in software design. These design actions meant that designers were choosing design alternatives that minimize the upfront investment in design without consideration of the costs of possible redesign work in the future. The design actions are summarized in Table 5-1.

Table 5-1: Design Actions, Description and Empirical Observations

Design action	**Description**	**Empirical Observations**
Reframing	Taking the requirements of the clients and users and reinterpreting them in different ways	Redefining, brainstorming, alternative solutions, reinterpreting
Simplifying	Creating simpler software design, such as by limiting the scope of existing design	Limiting the scope of the feature, keeping designs generic instead of customized
Coupling	Creating more tightly coupled software design	Workarounds, enhancing existing modules instead of creating new modules
Postponing	Postponing the investment in software design	Postponing until needed, postponing discussion, postponing until able to do it right

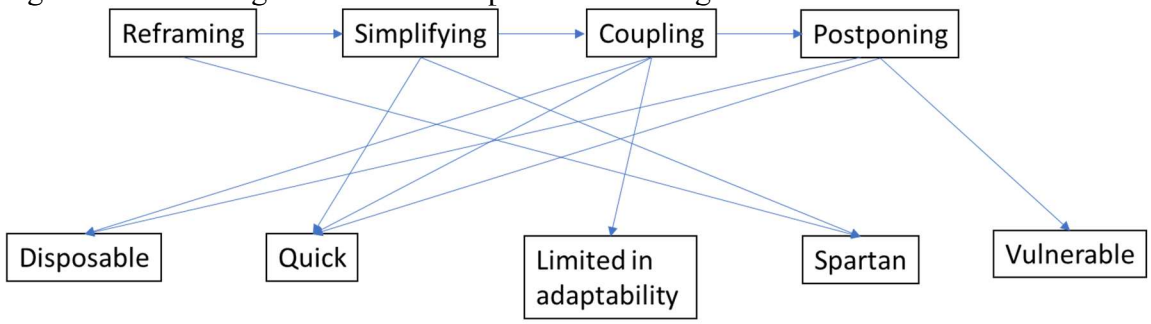
Table 5-2 highlights the key design characteristics of the digital product that differ from the typical product. As indicated in the previous section, the designs developed in the entrepreneurial context that are financial-focused need to be quick to build to enable the firm to capture the business opportunities.

Table 5-2: Characteristics of the Digital Product

Characteristic	Definition
Disposable	The product was short lived and meant to be discarded and redesigned in the future
Speed	The product was quick to design and deploy to clients
Limited in adaptability	The product could accommodate incremental changes but was limited in the flexibility to accommodate more changes or to scale.
Spartan	The product's functionality was limited in scope and sophistication.
Vulnerable	The product had high technical debt

Because of the design actions undertaken to help meet the clients' deadlines, the result is compromising quality, which results in designs that are disposable and spartan (with limited scope). Because of the compromises in quality, the design actions resulted in a digital product that was vulnerable. Figure 5-1 further summarizes how the design actions illustrated in the previous section contributed to these characteristics.

Figure 5-1: Design Actions and Impact on the Design Product Characteristics



It is important to note that the actions occurred in a linear fashion. When client requirements were received, the designers would first reframe the requirement based on their existing priorities, resources, and attention allocation. Thereafter, designers simplified

the design requirements to create a design product was just adequate for meeting the client's request. The designers also adopted coupling actions using workarounds, shortcuts and created more tightly coupled designs. These actions were undertaken to postpone investment of attention and resources to the future when needed.

5.1 Reframing Software as an Enabler of Funding

Because of the salience of financial-focused attention drivers, designers often reframed requirements in terms of designs critical for meeting revenue targets. By reframing, I refer to the action of taking the requirements of the clients and users and reinterpreting them in different ways (different forms of operationalization of design, different relationships between design components, etc.). Reframing was often the first step in the design process, as the designers isolated the mandatory components in the design requirements.

Reframing of the design feature was best exemplified in the design of the medication adherence feature. At the beginning of HealthCom's second year, the designers had just delivered the first version of the HealthCom software to the first two clients-investors. Shortly after, the designers' focus for the subsequent period, as determined by the investors, was to figure out how to grow the firm and identify the product market fit. As such, software was viewed as an enabler of funding. During this stage, HealthCom began to explore specialty pharmacy as a market segment to enter. After a discussion with prospective Client L, a specialty pharmacy client, the CEO ordered: "We will have to figure out how to do the medication adherence feature. This one comes from [prospective new client L], which has 60 clinics, 90k new patients a year." In view of the potential revenue from Client L, the Chief Designer rallied the designers to brainstorm ideas for the medication adherence feature. The CEO emphasized the potential revenue opportunity and

growth opportunities (i.e. the number of new patients) associated with this client. Thus, he believed that the designers should allocate attention and resources to design the medication adherence feature to capture this client.

Due to the investors' pressure to get more clients, the focus was, therefore, on designing software features that increased HealthCom's chances of a successful contract. The CEO suggested designing the medication adherence feature as a separate mobile application and expanding it to include a whole ecosystem of complementary mobile applications that he could then market to other clients. The iOS Designer was concerned with the interdependencies and extra work that would be needed. He responded, "Interesting idea. I need to give it some more thought... We already have so many apps that we have to maintain, and that number will rise with almost every new customer." Here, the iOS Designer was concerned with the resources needed to maintain a separate white-labeled app for each client to access the software. His concerns for quality was downplayed by the CEO's optimism for the opportunities the creation of a separate medication adherence mobile application could offer.

Interest in the design of the medication adherence was tied to the magnitude of the potential funding (i.e. a financial-focused design). Eventually, Client L did not sign a contract with HealthCom. The financial impetus for designing a medication adherence feature was removed. A few months before the completion of the study, discussions about the medication adherence feature resurfaced, as there was interest from another specialty pharmacy company. The resurgence of attention to the medication adherence feature highlighted the interdependencies between business opportunity and attention to design alternatives. Instead of designing a new mobile application for medication adherence, the Chief Designer decided to expand the scope of the existing task feature to accommodate medication adherence. The decision was prompted by a new designer who questioned why

the team would not consider, "enhancing the core infrastructure we have instead of building something entirely new." The Chief Designer and CEO reflected on this statement. They decided to reframe the medication adherence feature with respect to existing features in the HealthCom software. Designing the medication adherence feature as an extension of an existing feature would allow HealthCom to reuse code and reduced maintenance effort. Even though the prospective client was a big client, HealthCom's approach towards designing the medication adherence feature was less ambitious and resource-intensive. The designers reframed their conceptualization of a medication adherence feature as a type of task, which is already part of the design product.

The pressures to get the contracts thus became the impetus for design. It led to significant reframing of how designers viewed design requirements. Requirements were reinterpreted in terms of what would be necessary to achieve the design product promised to the client with minimal resources, instead of what the users wanted. For instance, the Chief Designer and iOS Designer had a debate about the design of the tasks feature, which was a core feature of the software. The iOS Designer heard from the users in Client-Investor C in his meeting, that they wanted a feature that allowed them to capture mood/side effects and a feature to administer surveys. Given that the current design product was the first version of the HealthCom software, the designers faced time constraints. They had to complete the design product on or before the deadline to prove that the HealthCom software was viable. Chief Designer decided to support these requests by leveraging existing designs to accommodate the new features. He created the appropriate user interface and reduced the scope of the requirements to support only a limited instance of the requirement. In response to a client request for the ability to track patient side effects (specifically mood) associated with their treatments, the Chief Designer reminded the iOS Designer:

IOS Designer: [after looking at the screenshot] I thought we didn't need side effects for v1? And the survey is undefined, too. Is it a checklist item? How do we get them to the website?

Chief Designer: Look, the ball is in our court on how to achieve certain functionality and prove value based on what our software will have available. If we're logging mood, we can effectively log patient satisfaction at the clinic. It's the same "task" with a different name. If we decided multiple choice checkbox is not going to be supported in 1.0, then we should remove symptom logger and postpone for version 1.0.

IOS Designer: I was under the impression that the survey would be a big web survey.

Chief Designer: No, I don't think it should be. I think Survey can be whatever the clinic wants, which is most often a 5-star like rating or something of the sort.

IOS Designer: Ah, ok. I just want to make sure we really iron out the version 1 requirements. We got 9 days, which means all those tasks need well-defined user interfaces in the app.

Here, the iOS Designer wanted to finalize the scope of requirements for the Care plan feature, since they had to release the software to Client C in nine days. He was, therefore, careful to make sure that all the design requirements were accounted for. In the screenshot the Chief Designer provided of the care plan feature, the iOS Designer saw that the Chief Designer had included an extra functionality (the ability to track side effects) that the iOS Designer thought was not required for this version of the software. The expansion of scope contrasted with previous examples in which the iOS Designer was faulted by the Chief Designer for trying to improve the designs. The iOS Designer believed that a survey feature would require a lot of design effort: he envisioned it to be a complex survey administration tool. The Chief Designer disagreed. He argued, "If we're logging mood, we can effectively log patient satisfaction at the clinic. It's the same "task" with a different name." The spartan nature of the survey feature contrasted with the software offerings of large patient satisfaction survey vendors that created software solutions for recording and disseminating patient satisfaction results.

By viewing the survey as a task type, designers could reduce the resources allocated to the design of the survey feature. Designers had the flexibility to decide how to design this feature because of the novelty of the feature: without explicit requirements from the clients about how the survey feature should work, designers could reframe how the feature should be designed. Any feature that could be adapted to include existing features was preferred. The ability to log patient satisfaction was constrained to the functionality provided by the task feature. The Chief Designer thus reframed the client requirement in terms of the design resources available in HealthCom, instead of imagining or understanding about how the user would use the feature. As the CTO explained, the ambiguity of the requirements of the novel feature led to the ability to reframe how the features were supposed to function. He said, “The conversation becomes one of "Is medication a version of this thing? Is it an extension of that or is it something different?" It's a reasonable question, right?” The CTO explained that new features could be reframed features and designed into existing features, by first conceptualizing the interdependencies that exist between the features to be designed.

Reframing also directed prioritization of designs in the process. In the discussion between the CTO and customer liaison, the CTO initially pushed back on the design request to create a “delete task” feature as not feasible. The intermediary at prospective Client K had provided feedback to the client intermediaries about the lack of organization in the task lists that the nurses had to manage. There were existing interdependencies that had to be cleanly resolved as an existing task would have existing data and interdependencies with care plans that would have to be severed if deleted. When the customer liaison articulated that it was a client request, the customer liaison and CTO worked together to examine the requirements and think differently about how they could achieve the requirements:

Customer Liaison: [To the CTO] This was a request from [our contact at prospective Client K]. It is a scale issue for any customer with a large population of patients. If you look at [Client A]'s task library, they could definitely use a delete feature. I think it's fair to propose we de-prioritize this. I'm guessing from your comment that this is a large undertaking.

CTO: [proposes an alternative of hiding the tasks instead of deleting]

Customer Liaison: I think that sounds like a reasonable solution! The issue currently is that the deactivated tasks stay in the task library making it not great to navigate when you have a large library of tasks. How about treating deactivated tasks the same way that we treat deactivated patients or archived messages? Just put them in a separate section of the dashboard?

In the above discussion, the Customer Liaison and CTO reached a compromise because she articulated quickly the reason for the design request, the problem it was posing, and value it could provide to prospective Client K and existing client A. The Customer Liaison reframed the problem into one was not simply a matter of improving quality and user experience, but one that had financial implications. She then worked with the CTO to devise alternative solutions, such as hiding the task to mimic the effect of deleting a task. By reframing the requirement in terms of the real need (i.e. to make the interface less cluttered for the user), they were able to devise workaround solutions that met the client's request. In determining "how" something should be built, the CTO often chose the least resource-intensive method. The CTO reminded the Customer Liaison, "Unless the [design requests] yields a bunch of money, you should kill it off." The financial returns associated with designs thus directed attention and resource allocation in design.

The CEO's emphasis on financial-focused design alternatives led to a different evaluation of alternatives. Rather than focusing on designs that improved the quality of the software or the user experience, his focus was on the ability of the software to help him

pursue business opportunities (i.e. “it would demo better”). It also meant directing the designers’ attention to design alternatives that may not be relevant to the firm’s existing clients. The CEO often recommending adding new features that made HealthCom appear technically savvy to prospective clients:

CEO: [To the iOS Designer] This list of features is more salesy/future-y, but is it in the realm of technical possibility?
iOS Designer: Health Kit maybe.
CEO: We should remove any aspect where we're talking out of our ass, but these seem like reasonable things our SDK should do. I'd like to leave Health Kit in if there's a shot it would work. It’s a nice value.
iOS Designer: User interface is doable but I’m not sure if it is worthwhile.

Recall that the demographics of the HealthCom software was the chronically ill, elderly population. It is not expected that their clients would have any use for HealthKit integration, which allowed manufacturers of smart fitness and health devices, and health and fitness apps, to share health data though the iOS mobile application. Throughout the observations, it became clear that the co-founders, especially the CEO, placed a high value on designs that improved the marketability of the HealthCom software to future clients, regardless of whether they would be used.

To conclude this section, the following illustrates how reframing of a design in terms of funding opportunities led to compromises in quality. HealthCom was designing the software development kits (SDK) for a major health IT client who wanted their IT software to be interoperable with the HealthCom software. The co-founders recognized that signing a deal with this client would open opportunities to other clients in the health IT market. Because of they were in the process of contract negotiation with this client, the co-founders only sought feedback from this client in the design process. The Chief

Designer argued with the iOS Designer to let the client choose from a list of SDK design alternatives, instead of deciding, as a firm, what was right for the HealthCom software. The iOS Designer was adamant about not doing so:

iOS Designer:	I don't want them thinking they have a choice to pick from all my options.
Chief Designer:	Their first pilot is 35,000 patients and they manage 9,000,000 patients. They can have all the choices they want.
IOS Designer:	It's our job to lead the [client] into making the right choice. Not letting them have so many choices that they can shoot themselves in the foot.

The belief of the co-founders was that designing the SDK according to how this client wanted it would increase the likelihood of a successful contract. Thus, the Chief Designer paid more attention to the client over the quality implications for the software. Additionally, the Chief Designer believed that because this prospective client, being a health IT vendor, was technologically savvy enough to make the appropriate design decision for HealthCom. As such, he did not see any threats to quality. He asserted, "We're going to be working with like-minded people. If they think like you do, then whether or not they have 3 options or 10 -- they should naturally agree."

The example highlighted the tension between the financial and quality-focused attention drivers that shaped ongoing design discussions. The decisions made in the design discussions were indications of the underlying attention driver for that designer. In the above conversation, the Chief Designer was focused on creating an SDK but how it was created did not matter to him. Conversely, the iOS Designer was focused on the quality implications of the design decisions. He asserted, "I just want to make sure we do what makes the most sense for us right now. This isn't a decision we should make lightly as it will have long-term effects on the company. Some of the SDK options we discussed require

massive amounts of short-term and long-term work." Instead of listening to the iOS Designer, the Chief Designer downplayed the concerns of the iOS Designer.

In the end, the client did not proceed with the contract. The designers later realized that leaving the decision to the client would have led to a less optimal design. For the HealthCom software to communicate with the client's dated software, the designers would have had to design plug-ins and code, which the CTO had thought to be a bad idea "n years ago." The client was not as technologically savvy as the Chief Designer had initially anticipated. If the designers had let the client choose the SDK design in view of the revenue opportunities, they would have been left with an inferior design. Even worse, the revenue never materialized but yet the designers would have to live with the repercussions of their decision.

5.2 Simplifying Client requirements

The goal of the software design, in the co-founders' perspective, was to be a version of the software that was adequate for meeting the client's requirements. They often believed that improvements to software design and quality could be made later. These improvements could be associated with additional revenue opportunities, as the CEO explained, "we will be spending a month or two that will be paying for our consulting time to help them make a really nice program. We might need to build a feature or two to make that better, but we will do that as we get there." Hence, attention to financial-focused design meant focusing on the elements of design that would be adequate for the client's requirements. The goal was to meet the deadline set by the client.

The CEO's dogma of creating designs that were just adequate for the client was echoed by the CTO. The CTO's role was to oversee the design work and allocate resources

among the designers on the various design tasks. As the following quote summarizes, his focus was on creating designs that could immediately demonstrate value for the firm:

> The idea is not to have waste, but I think about a lot in terms of always delivering things that are immediately of value at a level of business. ... if it is not, don't do that thing, whatever that thing is. ... if you are going to build something that is not going to deliver value right now, that is when you start going into these judgments about when it will deliver some kind of value.

The focus of the CEO and CTO on immediate value of design led to the drive for simplification in design.

Simplification of requirements increased with financial pressures. When HealthCom designed the very first version of the software for Clients-Investors A and C, instead of simplifying the requirements received, the designers sought to understand the needs of the users and clients. They sought frequent feedback from Clients A and C and their end users using mockups (screen shots) "to help [the client's] executive team visualize the solution and open up discussion" and to clarify any questions the other designers had with how a feature should be designed. The Chief Designer and CEO made frequent visits to the client sites, held user focus groups, and observed user workflows to understand issues pertaining to clinical workflow. The co-founders also encouraged their designers to take time to engage with the users.

As the financial pressures grew and HealthCom embarked on a search for product market fit, HealthCom's designers' attention became directed to creating a disease-agnostic software. They chose to create a disease-agnostic software so as to be more exploratory in their search for clients and increase the likelihood of successful contracts, given the long sales cycle in healthcare. Viewing software as an enabler of funding led to a simplification of the design requirements. The simplifying of requirements occurred in two main ways: (1) limiting of the scope in design and (2) the decision of generic instead of customized

design. As HealthCom's attention was directed to creating a disease-agnostic software, the co-founders began to raise concerns about the software being too customized to specific users. This was at odds with HealthCom's need to grow the startup quickly. When discussing the specifics of the feature design, the Chief Designer iterated the need to not be too concerned about these details requested by specific clients.

An example of simplification is the operationalization of a permissions model of who gets access to what patient information or user interface (formats, colors) that may be client specific. Instead of designing a permissions model that mirrored the organizational chart of a client, the designers chose a generic permissions model with nurse, nurse manager, and client intermediaries. The Chief Designer cautioned, The HealthCom mobile designs... aren't really supposed to be for one client or another. They are supposed to be reflective of a generic app unique to our software capabilities ... to service as many needs as possible." It was thus common for the Chief Designer to question his designers: "Is it a software-wide feature or client specific feature? If it's not supported on the backend, it's probably not a version 1.0 feature." As emphasized in the quote, attention and resources should be allocated to features that contributed to the software core designs that could be relevant to most, if not all the clients. Anything that was customized for specific clients that could not be marketed in the future to a more generic client was not to be included in this first version. For instance, in the previous discussion about adopting the tasks feature to accommodate side effects for Client C, the Chief designer reminded the iOS Designer, "Look, the ball is in our court on how to achieve certain functionality and prove value based on what our software will have available. If we're logging mood, we can effectively log patient satisfaction at the clinic. It's the same "task" with a different name." The argument was that this functionality could be performed by the existing task feature, and their expectations of a survey feature could be simplified (limited in scope).

These quotes highlight how the designers were cognizant that they should not be locked into the initial customer and should focus their efforts on what impacted the core software capabilities to ensure that the software would continue to function as an enabler of funding. The attention on generic aspects of the design requirements led to the design action of simplifying, primarily by only designing parts of the feature that would be generic and by limiting the scope of the design requirements.

Simplification also occurred when the designers chose to design for a subset of scenarios and use cases instead of all possible use cases. Instead of designing the software to mirror the complex relationships between diseases and therapies, between patients and their caregivers, or between types of nurses and administrators at the client sites, the designers often resorted to a simplification of requirements. The simplification of relationships in design also highlighted the difference in attention allocation among the designers, since they disagreed about how to model the initial relationships among the care plan, the patient's disease type, and therapy (drug treatment) received.

> iOS Designer: Is a Care plan linked to 1 Disease State and 1 therapy? The [existing data structure] coming back looks like it could have more than 1 of each since they come back as arrays.
>
> CTO: [To the iOS Designer] At present, it supports the possibility of being associated with multiple disease states and therapies. I haven't thought very deeply about that; I believe the consideration would be trying to get reuse out of care plans, so they don't have to be completely copied.

In this situation, it was easy for the CTO to create a more complex data table, since it involved creating extra columns and links in the supporting tables. The iOS Designer was, therefore, confused about what needed to be designed, since his understanding of the client needs, the CTO's design, and the screenshot provided by the Chief Designer for the user interface design did not match. The CTO explained that he did not think deeply about

exactly what the interdependencies would be. Although he had designed it to possibly support multiple disease states and therapies, it would not be needed for this first version of the software. He, therefore, advised the iOS Designer to simplify the relationships.

CTO: [To the iOS Designer] I wouldn't read too much into the relationships between disease states and therapies, which may turn out to be an overall weakness of the design. There are some things here we don't know.

iOS Designer: Well, I'm just trying to figure out how to model the app for version 1. We can change it later, but the more you change the model later the more than sucks since you have to write a bunch of upgrade code.

CTO: I think for the immediate future, it is reasonable to assume that the patient has one care plan. There is no constraint on the number of care plans associated to a patient. It is possible that there could be an unlimited number of care plans associated to a disease state. So, there might be 5 plans associated to [a particular disease state]. That is in scope for this release.

The iOS Designer questioned how the iOS application should be designed to support the looking up of a care plan for the patient. He explained that simply focusing on the present needs would lead to unnecessary redesign in the future. He pushed for the CTO to be more specific about the relationship between a care plan (the set of care instructions), the disease state of the patient, and the specific therapy prescribed for the patient. The iOS Designer argued:

iOS Designer: So what's confusing to me is the app has to have a reference to the patient, the "active" care plan, the "active" disease state, and the "active" therapy.

CTO: Yeah.... The backend has been designed for a more realistic world, where a patient might have multiple comorbidities, be in multiple treatment plans, etc. But, I think in the 1.0 release we're not really enabling any of that to be set up. The backend has no concept of "active" care plan, disease state, or therapy. The app and dashboard UI combine to constrain those things to "you can only choose one."

iOS Designer: Well, it kind of does, doesn't it?

CTO: But that's an artificial constraint. Only in the sense that you can only choose 1. That's initial state.

The Chief Designer and iOS Designer both disagreed with the CTO's simplification of the relationship and continued to question the decision. The CTO replied, "That gives me some anxiety, but they are sort of free floating now. What needs to happen, as I understand it, is that we need to build a cross walk between "disease" and the relevant care plans for the disease." The CTO's solution was to build a workaround that would enable the design to support multiple care plans, if needed, in the future. The CTO's recommendation was fo focus on what was necessary, i.e. to design the app and the web dashboard to support a constrained version to meet the client's deadline. The CTO thus advocated for a simplification of the design.

The simplification of requirements occurred, even if it could have implications for the reliability of the software. For example, the iOS Designer raised the issue of the mobile application crashing (failing to work) after a user tried to log in. He identified the issue as resulting from the nurse trying to enter a fractional "GoalNumber" (0.5 in this case). He then clarified with the Chief Designer about the client's design requirements for the task feature. Specifically, for every task assigned to a patient, the nurse has the option to prescribe a goal (such as drink 8 cups of water throughout the day). The existing mobile application only supported integers.

iOS Designer: Should the mobile application support fractional goal numbers?
Chief Designer: In the long-run, yes. For water, probably doesn't make sense but since the goal isn't unique to water, then should probably just make the water logger handle it on the client side.
iOS Designer: iOS currently truncates to integer
Chief Designer: I think you should probably stick with that for the water logger but I think that assumption could be dangerous if goal is used on other tasks, i,e. 'drink 3.5 oz of X medicine' and we decide to round up or down :) AAHH F*** IT, JUST DRINK 4 OZ - IT WON'T KILL YOU.

iOS Designer: Maybe. I hate GoalNumber tasks anyway. That would be a checklist task. Goalnumber is only important for long-running tasks that require multiple attempts to finish.

In the above conversation, the designers decided to ignore the complexities or requirements of healthcare and focus on what was adequate to deliver to the client by the deadline. They had no control over the types of tasks that nurses would prescribe through the HealthCom software, since the task editor was set up to be generic and open-ended. The nurses could potentially prescribe tasks for all disease types. Simplifying the design to only support integers limited the functionality of the task feature and also had dangerous repercussions for the patient. Even though the designers realized that the rounding up/down of the numbers could have detrimental consequences, such as in the case of medication administration, they chose to only focus on supporting integers in the first version of the task feature design.

In a final example of simplification, simplification could occur because of the novel nature of the feature. In the ShareMyCare feature design, the iOS Designer was wrestling with the idea of how permissions would be granted in this feature, in which patients can share their care information with a family member or friend:

iOS Designer: So, with the ShareMyCare feature, are the family members all granted the ability to complete tasks on behalf of the patient? or is that an extra permission that must be granted? [proceeds to clarify with Designer P how it is currently designed]

Designer P: I think the idea was that we have a requirement from Client B to support family relationships in general -- but that might not imply that they've agreed to share their personal health information with the family members. I know that's a separate step in a lot of health scenarios but I'm not sure to be honest -- CTO will probably chime in when he wakes up :)

While Designer P was aware that the situation might be more complex than imagined, he did not bother trying to understand the situation. The iOS Designer, on the other hand, questioned, and hypothesized alternative use cases. The iOS Designer sought to figure out

what exactly a family member would do with that read access to the HealthCom software. Would they try to input the task completion data on behalf of the patient on the software? For example, a son who helped his father with the medications may try to enter that information on the software on behalf of his father. The CTO, Chief Designer, and Designer P discussed, briefly, different hypothetical use cases. Because HealthCom was going to be the first among similar software to offer this feature, simplification was possible:

CTO:	Anyway, it's just to model that; may come up in future functionality. I think that's a good question. I might imagine that a family member would just see a list of the patients they can see, rather than doing a search, or something.
iOS Designer:	Yeah, it's a pretty different overall UI paradigm though
CTO:	Yeah. I think the general line of thinking is that "friends and family" might open some interesting new experiences on the mobile side. What those experiences are, I don't know how much anyone has thought about it.
iOS Designer:	Yeah, sounds like we might need a brainstorming session :)

In the initial stages of designing the ShareMyCare feature, the designers recognized that there was ambiguity and unknowns about how to operationalize the feature. Instead of engaging with the unknowns by brainstorming possibilities or questioning decisions, the attention was on the ability of the design to meet existing use cases, as opposed to imagined use cases or use cases with lower probabilities.

Overall, the exchanges above demonstrated the designers' use of simplification and their lack of desire to know what to design. Instead of being constrained by the unknowns, the iOS Designer embraced it and took it as an opportunity to brainstorm how to design a better feature. The CTO agreed with the need to understand how patients anticipated using the ShareMyCare feature but believed that this discussion could be postponed with no repercussions for the design.

In a less severe scenario, the designers also limited the scope of functionality delivered for each feature to minimize the effort required to create the feature, while meeting the overall requirements of the clients. For instance, in the design of the SecuredMessaging app, while the feature was supposed to allow patients and nurses to send file attachments and messages to each other, the iOS Designer suggested to the Chief Designer to simplify the requirements for version 1.0 of the SecuredMessaging feature:

> iOS Designer: For 1.0, I am leaving out file attachments. This includes audio and photos. Is that ok? Seems like we can add photos in an update.
> Chief Designer: I think it's okay, but we should definitely plan for it as a potential near-term feature
> iOS Designer: Sure. It feels like we need to find the smallest Minimum Viable Product we can, as this feature is super-complex.
> Chief Designer: I agree,
> iOS Designer: Ship quick and iterate

This is a typical discussion in which the designers, in attempts to meet the client's deadline, chose to reduce the scope of the functionality for the design product. Simplifying would give them more time to meet the deadline and allow them to learn more about the requirements for that feature over time, while limiting the upfront investment of attention and allocation. The typical conclusion to these design discussions was often that "we can revisit this later, if we find out that patients or users have the need for this." The decision was to reduce their upfront investment in the design process.

5.3 COUPLING DESIGNS

Coupling refers to the tightening of interdependencies between design components (such as modules and subsystems). Coupling actions meant a departure from designing the components as loosely coupled software (a best practice in software design). The HealthCom software comprised features, such as secured messaging or task editors, that were had to be accessible through organization-branded iOS or Android applications or

web portals. The features that are designed had to be supported on multiple platforms (e.g. iOS, Android, web), and thus, warranted the creation of modules and subsystems to support the features. The findings indicated that loose coupling was not always desirable among the designers, given their constraints. Instead, the findings suggest that designers adopted coupling in the design process to reduce the upfront investment of attention and resources in the design of specific features.

The tightening of interdependencies is best illustrated through the following two examples. In the first example, which deals with the medication adherence feature design in Section 5.1, the designers chose to design a feature not as a separate feature, but to expand an existing feature to include the new functionality. They decided to alter the existing task feature to accommodate a medication task type that could better support dosage complexities and enhanced (local) reminders on smartphones. While it required the rewriting of code in the software, the designers believed that this would improve the user experience for complex medication regimens and establish the foundation for alternative care plans in the future. Integrating the medication adherence feature into the existing task feature, would also reduce the design resources needed to create a separate medication adherence feature. It also had limitations for the future adaptability of the medication adherence feature, since it became constrained by the capabilities of the task feature.

In the second example, the designers were concurrently developing the SecuredMessaging feature and the ShareMyCare feature. The SecuredMessaging feature allowed nurses and patients to communicate through the HealthCom software, whereas the ShareMyCare feature allowed patients to invite other caregivers, such as family members, to view the tasks prescribed. In trying to figure out how to administer the user invitations functionality in the ShareMyCare feature, the CEO offered suggestions for a sufficient design for version 1.0 of the feature. He asserted, “Recall that we attacked this feature

because it was deemed to be low hanging fruit. Let's keep scope trimmed so our engineering ROI doesn't get lopsided. There's always time to make things better, especially when we have customer requirements driving it down the road. [Asks more questions about the feature] Perhaps [additional functionality] is something we postpone until version 1.0 is delivered." The Chief Designer concurred:

> Good feedback. I think your comment around Invitations may mean we should consider an Inbox-based approach like LinkedIn – which can be expanded upon later for other important messages or alerts… ..
> [Provides his feedback about the design alternatives]
>
> Separately, a conversation with [the CTO] about alerts and other software- level feedback drives me toward the Inbox pattern. Inboxes are familiar to almost every human being. It can allow someone to accept an invite or just let it linger - giving them more flexibility. … and can be expanded down the road as we support things like threshold alerts, messaging, etc.

The example showed how the designers saw ways in which SecuredMessaging and ShareMyCare features could be tightly coupled. By creating the inbox to support user invitations, the designers are, in effect, creating the architecture for an inbox for secured messages and other forms of notifications that could be designed later. The coupling of the features was a result of the designers' reframing and simplification of the design requirements.

The coupling of features together was often the simplified and quicker way of creating the features, while allowing the designers to meet the clients' requirements. However, by coupling the features, it limited the adaptability of the feature in the future. The CTO described:

> We built [the invitations in the ShareMyCare feature] in a way that was really a very traditional, [Create, Update, Delete] application. Just simple database calls, like really simple. We reached a point where, when we were working with [Client E] where they wanted the patient invitation feature to do a lot more stuff. The

> architecture we had for it, the design we had for it, however you want to think about it, was just not going to work.

Recall that the ShareMyCare feature was first designed for Client B. A later client, Client E, had different demands for the ShareMyCare feature that led to the need to rewrite the code. Consequently, the coupling that occurred upfront resulted in path dependencies in design decisions and limitations in combinations of modules in the future, since the functions of user invitations and secured messages became more tightly linked than necessary.

In an earlier example, simplifying the relationships in the design of care plans meant creating more tightly coupled software. The CTO's decision was to not "read too much into the relationships between disease states and therapies. Which may turn out to be an overall weakness of the design. There are some things here we don't know." Even though the designers made the relationships more tightly coupled than required, the CTO did not view the decision as problematic. The iOS Designer replied, "Well, I'm just trying to figure out how to model the app for [version 1]. We can change it later, but the more you change the model later the more than sucks since you have to write a bunch of upgrade code." Instead of discussing the potential typologies of care plans and disease states, the CTO viewed the consideration of the interdependencies as redundant and postponed it. The iOS Designer conversely recognized that coupling these relationships would limit the adaptability of the software in the future. It would create difficulty in separating the components or changing the designs in the future.

One of the key reasons why coupling was possible in this context was because of the malleable nature of interdependencies, given the novel design. The novel software requirements had ambiguous requirements and users. The interdependencies between the features was, therefore, subject to the interpretation and imagination of the designers. The

malleable interdependencies create opportunities for design, such as the ability to transfer costs across modules to allow the designers to overcome the constraints faced or to engage in coupling. For example, in the design of secured messaging and the management of user invites for ShareMyCare, the malleable interdependencies and ambiguous nature of the novel software enabled the designers to combine these two functionalities and to reuse code in the design process.

5.4 Postponing Investment of Time and Resources in Design

Postponing referred to the delaying of attention and resource allocation to design alternatives upfront. Postponing was frequently used in the HealthCom due to the time constraints, pressures to design features to meet revenue targets. The ambiguity in quality of product and novelty of the software also provided opportunities to postpone design, as the designers could deliver a product with lower quality and conserve design resources. Postponing also occurred, as the designers faced uncertainty regarding the future client characteristics in the search for a product market fit and thus did not want to over-customize their designs unnecessarily for any specific client type.

5.4.1 Postponing Until the Client Needed the Design

The first key form of postponing resulted from postponing investment until the design was needed to execute a contract with the client. For example, in the design of patient notes, which allowed patients to journal and record comments about the tasks in the software, the CEO suggested that they remove the function from the user interface for version 1.0. He justified, “Patient note is not actually a customer requirement at this point in time, and furthermore, we're not doing anything with that data for another 1-2 months because the dashboard isn't yet built.” Even though the functionality could be helpful to patients or nurses, there was no value to HealthCom in offering the feature. Therefore, they

chose not to invest resources in its design. The iOS Designer agreed, “Sounds fine to me, but in the end, this is the kind of thing that a client might demand, but until they do demand it, kill it.” The iOS Designer recognized the time constraints they faced with creating a working software for Clients-Investors C and A and agreed that the notes function would be of lower priority.

The postponement was often decided by the CEO, CTO, Chief Designer, or the Customer Liaison, who had frequent interactions with client intermediaries. They helped the designers with the prioritization which features to design now or in the future. For instance, she responded to the CTO and the Chief Designer, “Yeah, entering a name was not part of the [Business Requirements Document] nor the prototype [Client A] reviewed. I can see the argument for a checkbox. Otherwise, agree we punt until future.” Client A had signed a business requirements document stating that in exchange for increased revenue, the HealthCom software had to be able to capture the signature of a nurse making changes to a care plan, and/or confirm administration of treatment. In the design specifications, the Customer Liaison stated that having a field for the name of the nurse was not part of the agreed upon list, nor was it on the prototype (screenshot) that Client A reviewed and approved. As a result, it was acceptable to use a workaround solution for the time being, since it would allow them to postpone investment in design.

5.4.2 Postponing and Iterating in the Future

The second form of postponing was to delay the investment of design resources for now, with the expectation of reviewing the design and iterating in the future. For instance, in the following conversation, the designers were discussing the problems with the appointments and tasks features. Because the tasks were tied to the appointment date, these two features were tightly coupled. Patients often had to complete certain tasks pre-or post-

surgery or treatment (such as the appointment date). Even though in the testing stage, the designers were not consistently getting the right tasks, the designers decided to postpone attention and resource allocation to this problem:

> CEO: We'll sort this after version 1.0 is fully feature complete & stable.
> Chief Designer: Cool, the wormhole opened, and my future tasks came through.
> CTO: That's a much bigger discussion. Contemplate the reliability and maintainability implications of a client that can create the past or the future on its own.

Even though the design decisions had consequences for reliability and clinical implications for the users, the designers accepted a compromised quality of the feature for now, with the expectation that they would return to fix it after the first version of the software was fully completed. Even though the defect could require redesign in the future, the designer's attention was on the client's deadline and promised design product, while postponing allocation of resources to the design.

Postponement occurred even though there were threats to security, as well as legal ramifications. As highlighted in the earlier example regarding ShareMyCare permissions, the iOS Designer felt that the existing design of permissions was inadequate and exposed the firm to legal threats. The Chief Designer responded, "But we're not rewriting at this junction, that's my point.." He further challenged the iOS Designer, "and you're proposing what exactly? That we miss our [Client I's] 400 employee rollout and 8000 employee rollout Oct. 15th, to rewrite a problem that doesn't really exist at this junction?" The Chief Designer elaborated that the problems in the design were inconsequential at that point and would be best addressed at a future (unidentified) time. The Chief Designer, CEO, and CTO often talked about "starting with [something] and iterating" from there. For example, in response to the debate about the 'Share my Care' feature, the Chief Designer said, "If we need to better support the idea of relationship down the road, we can." Although the

Chief Designer was unwilling to make the modification, he acknowledged the iOS Designer's concerns and suggested postponing the change.

However, there was often little iteration. The lack of iteration was best articulated by the iOS Designer at the firm who said, "[The goal is to] get things out quickly, learn more about the feature and go back later and reiterate. But we never ended up going back to do things differently. The idea is that we potentially could." When I talked to the team about measuring use of the ShareMyCare feature, the CTO asserted that they did not think that use of the feature was high. There was thus no need to waste time analyzing use since it was a marketable feature. Therefore, the designers did not invest resources to better understand their users' requirements for the feature. With the financial pressures, there was a shift in the design approach to focus on minimal investments that generated revenue from emerging opportunities.

5.5 The Design Product

5.5.1 Disposability of the Design Product

Disposability referred to the short-lived nature of the product. The product was meant to be discarded and redesigned in the future. Because designs were viewed as disposable, designers often adopted coupling and postponing actions in the design process. The designers thus postponed investments in quality by using workarounds that did not resolve the problem and further shortened the longevity of the design. As the designers have described, they viewed the product as temporary software that enabled the startup to capture clients and grow its digital business. There resulted in a feedback loop.

The disposability of the software was also demonstrated by the example regarding the ShareMyCare user invitations that the designers created in conjunction with the SecuredMessaging feature. The designers chose to couple these features together. After

securing Client E, who eventually acquired HealthCom, they had to adapt the user invitation inbox to cater to Client E's requirements. As the CTO described:

> The existing design just wasn't very flexible. It would have taken a ton of work to make it work... Basically we would have had to rebuild a whole bunch of stuff we already had, features that actually existed [that] we would have had to build into the patient invitation functionality. It was easier to basically throw all the existing code away and redesign invitations as sort of an aspect of creating a patient.

Many similar examples reinforced the idea that the design product would be disposable since an investment of design resources could be redundant. If they had initially designed the user invites component with a more loosely coupled approach, it would have increased the flexibility of the code and reduced the need to redesign the software in the future. The problem was that this need may not have materialized. The previous design might have sufficed, and the investment of time and resources would have been wasted. Therefore, given the novelty of the design and the constraints faced by the designers, the decision was to create the product in a more disposable fashion.

While the design was adequate for the client, it was not intended to be robust or to scalable. In the case of HealthCom, it was expected that the design would be improved, or completely redesigned in the future. It was necessary for the HealthCom designers to design what the client wanted in the design product. As the Chief Designer contended, "Build it as cheaply as possible, put it out there. If it is bad, put it down. If it does work but maybe if you know that the reason is not working is for these reasons, fix it." The belief among the designers was that the design would not be long-lasting. The designers often actively lowered the software quality levels to what they felt was adequate for the clients. In this way, they did not have to commit more resources to existing requirements, in case they changed in the future. As the CTO described in an interview:

> Probably just with some boards, you can probably put together a room with a roof pretty quickly. It may be pretty hard to add another room…The cost is basically the upfront cost and the benefit is that down the road, you can move more quickly. The problem for a startup in particular is that you want to see that first room as fast as humanly possible. In some cases, you're willing to ... just throw the whole room away and start over again rather than spend the time to really plan out the room. Because maybe it turns out that your fundamental conception of the room was just totally wrong. If you don't know where you're going, it's really hard to build the right thing to get you there

As mentioned previously, the rules of the game for a startup dictate having to produce software quickly and to increase revenue. It was acceptable for designs to be thrown out and rewritten. If the conceptualization of the requirements by the clients or the designers were initially incorrect, this would avoid wasted upfront design cost. As the CTO explained, "[H]ow do we know when we are done? How do we know when this feature is right?" Ambiguity of requirements made it hard to evaluate quality, which could result in resources allocated to improving quality being wasted.

5.5.2 Speed in the Design Product

Speed referred to the ability of HealthCom to quickly design and deploy updated versions of the software to clients. Particularly, during the search for a product-market fit, HealthCom was in the exploratory stage. The flexibility and speed in responding to these opportunities was important, as opportunities, such as government reimbursement initiatives, emerged based on initiatives in the larger healthcare industry over the three years. The designers departed from designing customized solutions for specific client needs and instead viewed customization as inhibiting HealthCom's ability to market to different client types.

For instance, as the earlier example of the web dashboard for Client B showed, because the dashboard was conveyed as a deal breaker, the designers resorted to simplifying the requirements. They adopted a workaround to enable the designers to

leverage the existing nurse dashboard for the patient use case. For HealthCom, in the early stages, the alternative was to not design the feature and: 1) lose the revenue, 2) miss the opportunity to grow in this market segment in the future, and 3) lose credibility in front of the investor who introduced them to Client B. Being able to capture this new opportunity would increase revenue, which would reduce the uncertainty for HealthCom. It also increased the likelihood of HealthCom raising additional funding, since it made progress towards meeting investors' targets for sales and contracts. As a result, the designers chose to design the features quickly to capture the opportunity.

Because of the need for speed to capture emerging business opportunities, the designers adopted simplifying, coupling, and postponing actions that allowed the HealthCom designers to achieve speed in creating the design product. Simplifying requirements meant not having to design for the complex relationships that really existed, which increased the speed of creating the design. Coupling allowed for the reuse of codes as designs were more tightly coupled, which further increased the speed. The additional investment of design resources was also postponed until needed. These design actions led to a conservation of design effort and time and increased the speed in delivery of software to clients:

> The velocity for that is awesome, because any feature you're building can just reach in and touch any data and reuse any code and do anything from anywhere else. You get big ball of mud, spaghetti code, whatever. It has this advantage of huge velocity, but at some point, you realize that it actually hurts your velocity because you've reached the point where any change essentially is changing everything by implication…
>
> For a startup, it's probably closer to the big ball of mud. You don't really want to do that. You want to have it structured such that you can back out in the other direction as you get more time and resources, but you don't want to be so far over towards the highly abstracted curve that you can't onboard new people quickly, make changes quickly. Both of them get you to the exact same place, where you can't have velocity, but they happen for different reasons at different times.

The designers implemented processes that enabled the faster delivery of software. There was a preference for reduced discussions among the designers because greater participation meant lower efficiency. There was also a preference for fast, but a more tightly coupled and reduced scope, approach to design. As the CTO explained above, the tradeoff was velocity gains in the short-term versus velocity losses in the long-run due to redesign. However, the designers' focus was on firm performance in the immediate future, and often believed that the future redesign could be easily done with additional resources that could be hired from the funding received.

The preference for quick design solutions was illustrated by the designers' responses to defects or clients' requests. For instance, there was a last-minute workaround among the designers to meet an eleventh-hour requirement from Client A. Client A wanted their nurses to be able to retrieve the record of a patient without any categories or care plan assigned, so that they could assign them categories. The CTO checked with the iOS Designer about the existing implementation on the mobile application and the error that would result:

> CTO: [explains the workaround solution] That's currently not allowed in the software but is an 11th hour [Client A] requirement for Monday. Designer P is working on it.
>
> Chief Designer: I added a quick and dirty way we can change patient care plans and categories. this is on an individual basis and obviously not meant for changing groups of patients around

Because Client A wanted this requirement prior to their implementation for Monday, the designers were trying to figure out a solution on Thursday evening. Designer P was trying to figure out how to incorporate this requirement and create a reasonable user experience for the patient. The Chief Designer added a quick fix to the design to allow nurses to change patients' care plans individually, which would not work if the nurse had to manage a considerable number of patients, which would be the case for Client A. The function as

designed would cause inconvenience. On the surface, the quick workaround solution was adequate for being able to claim that it was possible for the nurse to make changes as requested, albeit only for one patient at a time.

5.5.3 Limited Adaptability of the Design Product

The cumulative design actions led to a design product that had limited adaptability. By limited adaptability, the product could accommodate incremental changes but was limited in the flexibility to accommodate more changes or to scale. The design products must allow the firm to be adaptable because of the emergent requests from clients as the digital business grows. As a HealthCom investor described:

> The CEO pivoted from their initial design because [of my feedback]. I've also brought him opportunities to talk with people that have ended up buying his product, and he's had to pivot again to meet the needs of that customer... they need the capital, and they need the credibility in the customer list.

To keep up with the clients' requirements and the investors' expectations for growth, designers had to constantly create new features or adapt the existing features to accommodate the requirements of new clients. The adaptable nature of software designs is an affordance of the general modular architecture of software. The designs can be expanded upon and coupled together to create new functions. The designers reused existing code and coupled to achieve other unintended functionalities. In the case of HealthCom, they leveraged the existing designs to expand the scope of features. For instance, in the case of the medication adherence feature discussed early, the designers chose to expand the existing task feature to include medication adherence. The existing code, to a certain extent, was adaptable and could help the designers meet client requests promptly. As the requirements grew with new clients, the decision to couple features together, as in the case of the medication adherence feature, could become limited. For instance, the Android

Designer said in response to the emergent nature of requirements in the design of the medication adherence feature:

> Even if you start out with a really nice medication adherence module, right? I mean the problem is inevitably you say, "Oh, great. Take one pill every two days." So, you just do one and two and that's the data you store: one pill, two days, every 24 hours or whatever. Oh, but this is an injection and then you say, "Well 300 milliliters every two days." And then someone says, "Oh but then you run out of needles every week and half."

While the initial interpretation of the designs might suffice, as the software becomes marketed to heterogeneous clients' new requirements may be received. The software becomes more tightly coupled as designers start to include new functionality in the existing software. As a result, it is not very modularized anymore because of the need for HealthCom to quickly meet client requirements.

Therefore, even though the design should remain adaptable to facilitate the capturing of business opportunities, in reality, the adaptability of the HealthCom software was limited. The software had been constructed in tightly coupled ways. Incremental changes to adapt to requirements were possible. Over time, changes, particularly bigger design changes, could become increasingly difficult because of the design actions undertaken when viewing the design as disposable. As the Android Designer described:

> I could write a fancy thing, or I could just jump in straight away. You always just go with the jump in straight away because you don't even know what's better at the time. But, inevitably what happens is that happens 25 times and all of the sudden you can't pull these two things apart.

Because design actions enabling speed were preferred, the result was for the designer to jump into the design, instead of thinking it through and creating a design that was more holistic and modularized. The decision to design more tightly coupled software meant that the design was hard to untangle. It was similar to the "ball of spaghetti" described by the CTO.

The following is another example that highlighted the limitations of the existing software design over time. When the iOS Designer, Android Designer, Web Designer O, and CTO were in discussion about the changes to the API and the possibility of breaking it into more manageable code, the designers varied in their responses. The Web Designer questioned how the APIs would be used and suggested that perhaps some of the existing code could be removed. The designers at this point found the patient end points to be complex and difficult to manage. It would be ideal to get rid of it. It would require major redesign on the part of all the designers given the existing design.

CTO:	At this point, we're probably stuck. If we could all agree that you must make a separate API call for anything that's different about a user versus a patient, we could make a transition...the problem I had was that they are so different you must do something ugly somewhere.
Web Designer O:	Ahh. That explains this pattern
CTO:	I'm open to proposals...truly. I chose to push the ugliness into someone else's code. I understand why this is not popular. Hence the openness to proposals.
Web Designer O:	[He evaluates the situation and concludes] So unless there is work yet to be done, I'd say leave it
CTO:	[Explains changes that would need to be done] If we could unify around a usable base concept for a "user" I'm pretty open to revision…sometime. Yeah... there are a bunch of things I wish were different. Unfortunately, none of them are the kinds of things the fixing of which will yield a nicer car for any of us.
iOS Designer:	Warm fuzzies > nicer car
CTO:	You say that now, when you can't smell the new car smell.

Despite recognizing that this current design was limited in adaptability and would not be sustainable (i.e. difficult to maintain) in the long-run, it was not something they had the resources to resolve at this present moment. Thus, the CTO postponed investment of design effort in fixing this code until a real need arose. As he said, fixing the code would not translate to any financial returns for him nor the designers, and was thus, not worth the time. To make sure that he and his designers did not waste time building things that did not

solve a real problem or lead to "a nicer car," the CTO acknowledged his constraints with the existing design and the resulting helplessness.

5.5.4 Spartan Nature of the Design Product

The design product was spartan in functionality, which meant that the product's functionality was limited in scope and sophistication. In HealthCom, the design product was always a simplified interpretation of the requirements. They captured the essence of the functionality required by the client but were limited in scope because they did not reflect the complex use cases that occurred. For instance, because of constraints (such as the availability of the designer), the designers could end up designing the feature only on the iOS and Android platforms but not on the web platform. For instance, the designers would limit the options on the feature, such that the design would work only for a limited use case, or if the patient clicked on the right keys.

> Chief Designer: So, what you propose is... if a patient suffers from Disease X and takes [Therapy Y], they should see the same [Therapy Y] treatment plan as the a [Disease Z] patient?
>
> CTO: My original proposal was that a patient with [Disease X] and [Therapy Y] *could* have a different care plan than a patient with [Disease Z] and [Therapy Y]. With the constraint that for now there is only one care plan for [Disease X] and [Therapy Y]. … At present, it supports the possibility of being associated with multiple disease states and therapies. I haven't thought very deeply about that; I believe the consideration would be trying to get reuse out of care plans, so they don't have to be completely copied.

These were artificial constraints imposed in the design, such as "one care plan for each combination for disease and therapy" that was not reflective of the reality of healthcare. However, designers often argued that limited options in the early versions of the design product would suffice. The design could be rewritten when the need for more complex use

cases arose. Simplifying and reframing the requirements led to a software that failed to meet the requirements of the user over time.

The consequences of the spartan nature of the design became evident when HealthCom managed to sign on three major clients, all of whom required the Chief Designer to rethink the software design. Existing use of the software was poor. By designing the features in a spartan manner without paying attention to the users and their requirements for the features, use of the software has been low. As a result, the spartan software design would inhibit the ability to sustain financial performance over time. The Chief Designer announced at the product meeting:

> [T]he next 3 customers are banner accounts for us. This is our cue to not screw things up. What is more concerning being that the whole thing is emphasizing patient engagement, which as you know is one of the hardest things and one of the things that we undermine it the most, given that we are a sales-oriented organization ... And so, moving forward, 100% of my time will be dedicated to product management and understanding utilization of features and trying to do better at just specifying features in general.

The Chief Designer acknowledged that to date, as an organization, the focus had been on meeting the funding-related goals. Now that they had signed on three major clients, the problems related to the quality of the software and the actual design flaws needed to be addressed for the funding to materialize. As seen in the earlier sections, the focus was on increasing the number of features, as opposed to attention on how the features are supposed to function, as the designers often reframed and simplified the requirements to minimize the attention and resource allocation upfront. HealthCom's focus on designing a plethora of features reflected the intention to quickly capture revenue-generating opportunities and to gain access to more funding with the caveat that in the future the software would be improved.

The attention of the designers was not on eliciting accurate user requirements for software, but in creating many features with spartan functionality that did not work as well. The cofounders recognized that the poor use levels of Client D and B would not be able to withstand the scrutiny of the bigger clients once they began using the software. The feature needed to produce real-life improvements in healthcare delivery and changes in patient engagement. These major clients would be evaluating the HealthCom software on quality-related metrics. Given the reputation of these clients, performing well on these metrics would improve the marketability of the HealthCom software. Their lack of desire to invest resources and attention upfront to these elements of design meant that they were now faced with spartan designs that did not really meet the needs of the user.

5.5.5 Vulnerability of the Design Product

The design products were vulnerable. The product had high technical debt. Because of postponing the investment of design resources, as highlighted in Section 5.4, the software designed accumulated technical debt that was not addressed. Compromised quality decisions and tightly coupled interdependencies meant that there was the need for future redesign. For example, in the HealthCom case, every deployment of the designs to the production server was always defect-ridden. The designers were up to midnight or the early hours of the morning trying to resolve defects and error messages because the components that were designed were not compatible.

To meet the financial-directed deadlines, the designers compromised quality, for example, by employing interns and contractors. The designers knew that the decisions compromised the quality of the final design. For instance, in the design of the web dashboard, the CTO and Chief Designer engaged contractors in order to get the design done quickly. The patient web dashboard was a requirement of Client B, and as the CTO

said, “The original architecture of the dashboard is not that good, ... You know we hired a contractor to build it. I don’t think it’s built as well as it could have been built.” Part of this was also because the CEO, Chief Designer, and CEO did not believe the web dashboard was a feature the clients needed given their goal to target large clients with their own health IT software. The following is a discussion about the web dashboard defects and the resources that were needed to fix it:

CTO: We also have a lot of technical debt in the dashboard
CEO: Is that the fault of the contractor guy?
Chief Designer: I think most of that got worked out and it’s just [the contractor] not being an expert on [a particular aspect of web design], but he just trying to make things work.
CTO: So that will be yes
CEO: How much did that actually cost us? We got a sweet deal from that guy, so I’m just wondering if we are on the upside of our investment?
Chief Designer: It got us where we needed to be.
CTO: It’s hard to evaluate that right, because we needed it done and he was our only option. But certainly, it’s added a couple of months of work for [Web Designer O] at least.
Chief Designer: But I wouldn’t say that it’s [the contractor’s] fault, would you?
CTO: But he wrote the thing
Chief Designer: Yeah but…

In the discussion above, the CEO was focused on trying to determine the appropriateness of the decision of the quality compromises. Quality or financial implications of the decision could not be evaluated post hoc because the alternative was not getting Client B’s contract. Although the contractor was not as skilled at web design, he was the only resource they could find on a short notice to design the web dashboard for Client B. Therefore, it was hard to figure out if the quality compromise was worthwhile because they had to look to the short-term to meet the client’s deadline. As the CTO said, “In the end, the thing you can deliver now is probably worth more than the thing you deliver in 6 months and wait for it to be perfect. It’s one of those things that is different about [selling to] enterprise.”

The CTO accepted the fact that having to sell to businesses and meet investors' expectations meant focusing on creating design products that met the clients' deadlines and requirements. The CTO explained how the web dashboard was a bootstrapped design that was put together at the request of Client B. They had not intended it to be patient facing. It was supposed to have been a temporary solution for nurses to configure the care plans until they could be integrated into the healthcare provider's electronic medical record. As a result, the web dashboard was a hodge-podge of code that did not function well together.

The focus on the designs that would improve financial performance drove decisions to focus on new features that would increase revenue generated from clients. The tension was articulated by another platform engineer, who said, "Generally, [these backend technical designs] stay in the low priority bucket until they become critical." Prioritization of resources for iterations and integration was not observed across the data sources of observations, interviews, or chat transcripts. For instance, as one of the platform designers confirmed:

> So, the site will be slow, or [the nurses] won't be getting their notifications or something along those lines. Meanwhile, alerts and emails and all kinds of things are firing off into our inboxes. We're staying up night after night trying to figure out how to solve the problem. That's usually how it sorts of gets from, "Well, we sort of anticipate it a little bit." to then all of the sudden, the sky is falling, and we need to fix it. That's been true almost every startup I've been at where you sort of know when you build something that it has a certain life span in it. Yeah. It's always a balancing act between "do I build this for tomorrow" or "do I build this for today?"

The other issue, as mentioned previously, was that investment in resources in the reduction of technical debt could not always be justified given the need to design features to meet clients' deadlines. For example, the CTO said,

> You got to start thinking about whether, "Do I care [about the slow performance]? What about this other thing that takes 2 seconds? It's not terrible, but it's not great. You know. Is that thing now that is 2 seconds – when I have 1 million

> patients. Is that still going to be 2 seconds? Or is that going to be 200 s? So, for e.g. the dashboard, we know that it's broken [ridden with performance issues]. But we don't have resources to do anything about it.

As described in the mechanism section, the preference of many designers was to adopt quick fixes or "band-aid" solutions that resolved the defects symptomatically but did not get at the root of the problem. Cumulatively, this resulted in technical debt over time. Instead of focusing on eliminating technical debt, the focus was on the "shiny features" described by the Platform Designer L:

> Shiny [HealthKit] didn't have anything to do with our customers needed. It made us look progressive. That's just an example of [something] that helps with marketing. It's something that makes you look more presentable. If you think on a scale of functionality versus technical performance, frequently on a feature functionality takes primary and performance takes secondary place.

Consequently, flashy designs and new features were prioritized. They did not contribute to the improvement of software performance. For instance, when there was poor use of the software among the nurses and patients at Client D (terminated contract) and Client B, the designers were not interested in understanding the root causes. The Chief Designer contended, "One might argue it has to do with our app, but I would argue it has to do with the content [generated by the user]." Instead of trying to fix the content or enhance existing features, they built a secured messaging feature in attempts to increase software use. The designers believed that building a messaging app would encourage nurses and patients to communicate using the software. Software use continued to fall. For instance, with Client B, application use fell from 100% to 5% in the first ten weeks.

5.6 EPILOGUE

To provide insight into the outcomes of the firm, I closed my time in the field with a series of interviews with the external players. I asked these participants to articulate their thoughts about the HealthCom business and design product. I used this exercise to spark a

broader discussion of the financial-focused designs that would become an avenue for future research.

5.6.1 Difficulty Getting Contracts with Larger Clients

With the design product, it was difficult getting contracts with larger clients with more stringent evaluation criteria for software quality. For instance, many of the larger healthcare providers had a due diligence process during the contract negotiation during which their IT teams would visit HealthCom to evaluate the quality of the design code of the software they were licensing. One of the major health IT firms interviewed said:

> We felt, after evaluating their code and what was actually available in the product and the number of resources they had implementing it, that they were a little top heavy on management. They lacked the design resources and that there was a good percentage of the code on that front end that would need to be rewritten in order to conform to a larger audience.

In this conversation, the client alluded to the difficulties they would face and the work they would have to invest in updating the code to make sure that it would be able to serve the hundreds of thousands of users the health IT firm was serving. Being able to license the software to a major client would be a boost to the funding goals. However, the existing design product made it impossible to develop software that would be good enough for these clients to license.

Another prospective HealthCom client similarly recognized the constraints of working with a growing startup that faced funding goals. The client spoke to the challenges of integration for themselves, “It's not like it's all win and no cost. Besides [losing control over] the patient experience, you become less flexible with your own road map. All these factors lead us to make the decisions.” There were costs associated with trying to integrate and work with HealthCom because it was a software that patients would be using and representing the client’s name. Therefore, the combination of losing the control of the

patient experience and watching HealthCom defer to other clients in the pursuit of business opportunities, did not bode well. As the client explained:

> Basically, HealthCom said, "Oh, we'll do whatever you want." What I meant is, let's say, for us maybe it wouldn't have been that big of an issue because we were probably the biggest client of HealthCom, but let's say they got another really big client. They got another really big client and that big clients starts saying they should start dictating the road map... At this time, they've taken the road map and that road map basically doesn't match with our road map. It doesn't cover the use cases that we want to cover. What are we going to do? We're already invested in integration. We already pushed all this application and in the launch. To our patients. That's the cost to consider. What's going to be my switching cost?

The client saw how willing HealthCom was in deferring to the client to meet their requirements, as it was a publicly listed firm. If HealthCom were to capture this business opportunity, it would alleviate HealthCom's pressures from their funding goals. It would also help HealthCom to sell its software to the other clients given the credibility gains. The co-founders' deference to the client also worried them. They recognized that they were losing control over the patient facing version of the software, and they might also lose control over the design process time should HealthCom work with another bigger client.

5.6.2 Technical Challenges in Design

As the conversations throughout the chapters have highlighted, the design product created technical challenges in design over time. Because of the quality compromises and coupling actions, the designers often had no idea how the interdependencies in the software had been designed. Coupling increased technical debt and defects that were difficult to troubleshoot. For instance, the night of the deployment of the software, there were major defects that would hinder the implementation at the client site the following day.

CTO:	Basically, activation was fully and completely busted, among other things. [Describes the problem] Definitely not our finest hour.
Chief Designer:	Well I think this is representative of a lot of misjuggling

CTO: Yeah, a lot of distraction and context switching definitely contributed on the engineering side, and I'm sure for other people as well.

Chief Designer: All good problems we need to figure out but not ones we WANT to have as professionals. [Complains about testing vendor they have been using]

Chief Designer: I agree with resource constraints but want to also make sure we don't over-correct. At the end of the day, we haven't had a lot of breathing room to test *new* features. [Client C] is now (after months) up to just 72 patients and [Client A] barely scratches 50 patients.

CTO: That's true; but we're building a product that can accommodate vastly larger numbers in a more and more complex software.

Chief Designer: [Client B] has asked for a lot, to deploy to a lot, without a lot of breathing room. I agree

CTO: I think we're doing the right things - adding an engineer, maybe an implementation person. I don't think we have the signals for much beyond that yet.

The impact in the software was a "adequate for now" design that was not necessarily coherent nor robust. In the discussion above, the CTO admitted his mis-juggling of the design work that was exacerbated by the constraints and Client B's deadlines. The QA vendor they engaged also failed to detect all of the quality issues. The user engagement rate was also dismal - Client C had only 72 patients and Client A had only 50 users. The low user engagement rate also led to the reduced desire to invest design resources to design for the robustness that was needed for a bigger user base. The Chief Designer believed that more features would resolve the problem of low software use, without understanding the reasons for low use of the software. The conversation also reinforced how the Chief Designer and CTO believed that the software quality could be addressed when the firm had more resources.

The CTO also recognized the challenges of the technical limitations of the existing software. As the client of the chief health IT described, it would require major redesign. Similarly, in response to the interest from another major client, the CTO said, "They were

pretty heavy on the enterprise data architecture front. Being on top of our game there is going to matter." He recognized that the weak link in the design of HealthCom was the enterprise data architecture. It would require more resources to ensure that it met the specifications of the larger firms.

5.6.3 Impact on the Organization

Platform Designer L revealed that he did not feel that HealthCom was going to succeed. He elaborated on the difficulty HealthCom experienced in getting Series A funding and how they could not demonstrate the product market fit that the investors wanted. The iOS Designer and Android Designer also left shortly before the end of my study. The iOS Designer made significant quality compromises in the design process to meet the deadlines. He said:

> There wasn't a ton of discussion. [The Chief Designer] mostly decided what was worked on, what would be in each release…Occasionally, [the CEO] would bring in requirements from client calls... To be brutally honest, one of the reason I chose to [leave HealthCom], Chief Designer controls process and would not let us own more of the product and drive product decisions. There were moments I requested more control – Chief Designer said I was not meeting with clients, he should hold the control.

Throughout the discussions, the iOS Designer's perspectives were often not respected. In the end, the iOS Designer decided to move to another startup, where he felt that he could have more voice in the design process.

5.6.4 Acquisition

Despite the quality concerns that were highlighted in the earlier sections, one of the intriguing results was that HealthCom was acquired by a major health IT firm eight months after the termination of the study. The acquisition is the happily-ever-after that many

startup entrepreneurs desire. The CTO credited the sale of HealthCom to his design of the software architecture:

> In a startup, I think good software architecture almost always takes a back seat to rapidity of design and ability to change quickly. I think in a lot of ways, HealthCom was over-engineered. I would argue that probably I just too much energy into making a pretty good architecture from the beginning.
>
> On the other hand, had I not done that, [Client E] never would have bought us. To some extent, they were actually buying that architecture. Probably while it was more robust than extensible and clean, particularly initially, it slowed our total velocity. I think that the path we were going down, that going down it faster probably wouldn't have helped. And it would have hurt, in a sense, because I think that we actually would never have been acquired.

Retrospectively, the CTO believed that the architecture helped facilitate the acquisition, although it slowed down the initial growth and design of the software. The comment was intriguing, given knowledge of how the software was designed. Two weeks after the termination of the study, HealthCom signed contracts with two major clients with whom they were in negotiations throughout the duration of the study.

5.7 SUMMARY

In this chapter, I explained the design actions: reframing, simplifying, coupling, and postponing. Subsequently, the design actions culminated in a design product that was disposable, quick to build, limited in adaptability, spartan, and vulnerable. The characteristics were a result of design actions to enable the firm to capture the emerging business opportunities and improved financial performance. Finally, I closed the chapter with an epilogue to provide insight into the outcomes of the firm after the termination of the study.

Chapter 6: Discussion, Implications, and Future Research

6.1 Overview of Chapter

The goal of this dissertation is to understand the research question of: How does the entrepreneurial context in which the startup operates affect the software design process and product in the early years? In this chapter, I briefly summarize the major findings from the dissertation and triangulate them with the existing literature. I discuss some of the theoretical and practical implications that can be drawn from this research.

6.2 Impact of the Early Stage Entrepreneurial Context on Design Actions

The findings suggest that the entrepreneurial context influences attention allocation amongst the designers and create tensions between financial and quality-focused designs. Time and financial constraints intensify the tensions between financial and quality-focused designs. Additionally, the search for the suitable market in a tight time frame throughout the early stage entrepreneurial context renders the financial-focused players critical (e.g., Nambisan, 2017; Antonopoulos et al., 2014). The salience of the financial-focused players and rules of the game have resulted in a prioritization of financial-focused designs over quality-focused designs.

These findings extend the existing literature on software design in entrepreneurial firms. Yet, the focus of existing research on design in entrepreneurial context has not been examined the impact of attention to financial-focused designs on the software design actions and product. Antonopoulou et al. (2016) has examined the use of repurposing, i.e. modifying and reusing, code for different business contexts to increase the startup's ability to capture emerging business opportunities. However, the authors do not describe the tensions designers face and instead present repurposing as a panacea for designing features

quickly for product market search. Repurposing is afforded by the modular architecture of software, but as my findings indicate, the designers tend to create more tightly coupled designs, which would make repurposing challenging. So how can designers in early stage entrepreneurial contexts adopt coupling in ways that provide the flexibility needed to support business model exploration?

My findings also highlight a departure from existing software design practices. The software design literature maintains that a typical design process starts with requirements analysis, design, testing, maintenance, and revision (Royce, 1970). The proportion of time spent in each phase varies, depending on whether it is a plan based or flexible approach (e.g. Beck 2001, Royce 1970, Sommerville, 1996). During the analysis phase, designers adopting the Waterfall method seek to elicit documenting requirements accurately (e.g. Royce, 1970, Sommerville, 1996). My findings show that in the early stage startup, designers reframed and simplified the requirements based on the clients' requirements and the designers' choices. In the design phase, designers using the Waterfall method would map the architecture (Royce, 1970), or design prototypes for feedback and analysis in the Agile and Lean Startup method (e.g. Beck 2001, Ries, 2011). Conversely, designers in the early stage startup adopted coupling and postponing actions and paid little attention to software architecture. Lastly, the findings highlighted lack of revision and iteration in the early stage startup. The lack of iteration differed from the revision phase in the Waterfall method and extensive feedback and learning conducted in Agile and Lean Startup methods. The design process in the early stage entrepreneurial context thus suggests extreme minimality in the design process.

Existing research on Agile methods has stressed the lack of rigor in application of software design practices (Conboy, 2009; Fitzgerald, 1997). Examining how designers in an early stage entrepreneurial setting depart from design practices provides insight into the

design actions are undertaken and the rationales for adopting or departing from the espoused practices.

6.3 Impact of the Early Stage Entrepreneurial Context on the Design Product

My findings suggest the creation of an alternative design product in the early stage entrepreneurial context. My findings show that attention to scale and robustness in designs were not perspectives that were rewarded in the time-constrained, entrepreneurial environment. Rather, they were viewed as over-optimization of design quality. Conversely, designers viewed compromised quality as a necessary evil in their pursuit of emerging business opportunities. These characteristics have been identified by Elbanna and Sarker (2016) in their review of designs completed using the Agile method. Elbanna and Sarker (2016) find that the short design cycles and prototypes in Agile methods lead to cumulative technical debt. However, Tumbas et al. (2017) suggest that early stage startups have little technical debt. This contradicts my findings which show that designers compromised quality and postponed investments in design that lead to accumulation of technical debt.

To enhancing the understanding of the impact of the early stage entrepreneurial context on the design product, I propose the notion of a digital exoskeleton. I will offer my conceptualization of the digital exoskeleton, before delving into the similarities and differences between the digital exoskeleton and other design products studied in existing research.

6.3.1 The Exoskeleton and Parallels to Biology and Software Design in Entrepreneurial Contexts

Biology defines the exoskeleton as the external skeleton that supports and protects an animal's body ("Exoskeleton," n.d.). Like the design product, exoskeletons are rigid

and present some limitations to growth. While some organisms can add new materials to their shells, exoskeletons often must be shed or molted when outgrown by the arthropod. Examples include hermit crab that moves into a new shell when it outgrows the old shell or the caterpillar that undergoes a metamorphosis transformation into the butterfly.

Parallels can be made with the phenomenon of software startup building a software and digital business concurrently. The software created by a digital business startup is akin to the exoskeleton, where the arthropod is the digital business (startup). As the digital business grows, so does the exoskeleton, but only incrementally. The digital business can add new features to its software, but as the findings indicate, the design actions have little consideration for software architecture and result in a tightly coupled software with limited adaptability.

In the case of a startup, as the digital business unfolds and grows, it reaches a point where it outgrows the existing software (i.e., the software becomes too rigid for growth). Redesign of the software is needed. The need for a new design has been observed and documented in the practitioner literature (Tran and Zhu, 2016), in which evolution of the business model led to the need for redesign of the exoskeleton. Information Systems scholars have alluded to the need for redesign and changes in design because of the technical debt accumulation in the use of Agile methods (Cao et al., 2009; Cao et al., 2010), but the impetus for change in design does not arise from business model changes. The evolution of the digital exoskeleton is linked to the design of the digital business. The startup must hence shed its existing exoskeleton (i.e., design code) to grow its digital business.

While the exoskeleton metaphor does not translate perfectly to the software design context, what the startup is building is a digital exoskeleton that is deployed to users. The designers see the digital exoskeleton as short lived. The findings highlighted the designers'

expectation that the existing product would be redesigned when they had more resources or were required to scale the business and software. The designers expected to build a new, larger exoskeleton or, in an extreme case, a complete metamorphosis may have been needed to achieve more drastic strategic changes.

6.3.2 Value of the Digital Exoskeleton

Designing software as a digital exoskeleton helped the designers meet their deadlines. By designing software as disposable, spartan and with limited adaptability, the designers could build the design product quickly. The problems with working with a design with limited adaptability and high vulnerability were redesign costs incurred in the future. The digital exoskeleton reflects a design process that is focused on the short-term financial performance with disregard for the impact on quality, as short-term firm performance and survival are paramount to the entrepreneurs. Like the exoskeleton that protects the arthropod from threats to survival ("Exoskeleton," n.d.), the digital exoskeleton offers the startup protection from failure, by serving as an asset for marketing and revenue generation.

The existing literature in the software design recommended the use of prototypes and MVPs in uncertain contexts. Both of these design methods focus on the design of separate features in each prototyping cycle, but do not examine the impact of uncertainty on the design of the product as a whole (e.g. Elbanna & Sarker, 2016; Sharkey, 2013). As such, like the digital exoskeleton, MVPs and prototypes have high technical debt and are thus vulnerable.

To that end, the digital exoskeleton deviates from the current understanding of prototypes and MVPs, in its origin and perception to users. The prototype and MVPs are designed for the entrepreneur to learn about the users (Eisenmann et al., 2013; Ries, 2011) so as to create a product of value to the users. The findings suggest that designers in the

early stage entrepeneurial context are not using the digital exoskeleton to learn about the users and their requirements. Instead, it is used by the startup to increase its own value.

Additionally, for the user, prototypes are interim products used for feedback and communication (e.g. Beck 2001) but the digital exoskeleton is viewed as a working software that did not appear to be a temporary software to the user. In HealthCom's case, sensitive patient information was communicated on this vulnerable, disposable digital exoskeleton. The vulnerability of the software raises concerns about the tension between quality and novelty that could be addressed in future research.

6.4 Implications For Theory

6.4.1 Attention Allocation and Design Actions in the Early Stage Entrepreneurial Context

I contribute to the ongoing discussion on software design. Scholars have pointed to the interdependencies between design of software and digital business in the entrepreneurial context (Antonopoulou et al., 2014, 2016). I highlight a critical difference in the salience of the digital business rules of the game for designers in early stage entrepreneurial contexts and how it impacts the designers' attention allocation and actions.

Firstly, the influence of the investors in the design process was intensified in the early stage entrepreneurial setting. Although investors were not directly participating in the software design discussions, their influence could be felt through actions, such as the withholding of subsequent funding.

Secondly, existing research has pointed to the differences in designers' attention allocation depending on their roles in designs (e.g. Sawyer, 2001; Lyytinen and Newman 2015). These roles influence what designers pay attention to and actions they take in the

design process (e.g. Austin, 2001; Abdel-Hamid et al., 1989; Lee and Xia, 2010). For instance, Sawyer (2001) find that even among designers, they can have different and divergent goals that vary across roles. These goals create conflict among the designers and the differences are difficult to reconcile (Sawyer, 2001).

The tension is intensified in the case of designers in entrepreneurial firms who face different attention drivers. Unlike the Sawyer (2001) study in which the designers adopt single roles, designers in the early stage startups may take on multiple roles (such as entrepreneur and software designer). The multiplicity of roles drives the difference in salience of attention drivers and affects the designers' attention allocation and action.

The designers' limited attention is pulled between the different attention drivers. In large organizations, although designers face budgetary or schedule constraints (e.g. Austin, 2001; Abdel-Hamid et al., 1989; Lee and Xia, 2010), the work of the design and business units are separated. While uncertainty may persist, uncertainty for designers stem from design process uncertainty (Lyytinen 2001; Tuomi, 2002), such as the ability to meet deadlines and stay within budget. Conversely, designers in startups face uncertainty with regards to the product, product-market, design process and survival of the firm. Similarities to the early entrepreneurial context may be observed in new product innovation teams in large firms, but these design teams are often isolated from the rest of the firm or sheltered from the uncertainty with the resources of the parent firm. In contrast, in the early stage entrepreneurial context, the designers acutely feel the financial pressures and urgency to generate designs that improve financial performance. Uncertainty is heightened particularly if the firm is also operating in a nascent market (Santos and Eisenhardt, 2005).

My findings highlight actions designers have undertaken to work through the tensions. The study did not examine the goals of the individual designers and their motivations for joining a startup, which may affect how they allocated their attention in the

design process. For instance, some designers may not be interested in making quality software at all but instead using their skills to make money. Designers in the entrepreneurial setting might differ in their design approaches or expectations of quality from those in software design teams in large firms. Hence, designers in entrepreneurial settings may be more likely to make quality compromises in design.

The discussion on attention allocation, tensions and adopted design actions can benefit from an understanding of the various intrinsic and extrinsic motivations of designers in entrepreneurial settings.

6.4.2 The Changing Relationship Between Software Quality and Firm Performance

Adopting the ABV in this dissertation allowed me to systematically analyze the attention drivers that originated from various levels of analysis (e.g. organizational or environmental) (Ocasio, 1997). The external, internal, and economic factors were observed to affect the designers' attention allocation. While the ABV has shed light on the relationship between the design efforts, puzzles remain on the relationship between software quality and firm performance.

Previous IS research has emphasized the need for software designs that meet user requirements (Guinan et al., 1998). The pursuit of designs practices that facilitate the process of designing for users have motivated research on ways to improve the requirements elicitation process (E.g. Iansiti and MacCormack 1997; Banker and Slaughter, 200; Guinan et al. 1998). The need for designers to pay attention to the users have also catalyzed research on software design approaches (e.g. plan-based versus controlled) to streamline the design process and account for contingencies, such as task complexity (e.g. Maruping et al. 2009).

Yet, the creation of spartan functions, vulnerable product and lack of user-centeredness in my findings provide a stark contrast to the research in software design. How do we make sense of the relationship between software quality and firm performance? This dissertation begins to highlight the designers' locus of attention throughout the design process. When faced with tensions between quality and financial-focused designs, the designers' attention was on the latter. Only in the case where an inferior design product would inhibit the pursuit of business opportunities would attention be redirected to quality-focused designs.

The findings begin to suggest an alternative view of software design as a means, instead of an end (Antonopoulou et al. 2016). Given the attention of the designers on revenue target, the software becomes an enabler of funding and thus effort has been directed into designs that help designers meet that goal. It likens the designer to a Pied Piper of startups, who leverages the design product and its existing portfolio of clients to attract more and bigger clients.

The findings also suggest the need for researchers to consider redefinition of metrics upon which the quality of the digital exoskeleton should be defined in the early stages of a startup's life, if the digital exoskeleton's purpose is to help the startup achieve revenue targets.

6.4.3 Leveraging Chat Transcripts for Insights into the Micro-Processes in Design

Although research has begun to focus on the work of startup entrepreneurs (e.g. Tumbas et al., 2015, 2017; Antonopoulou et al., 2016), this dissertation is one of the first that provides fine-grained insights into the decisions and discussions amongst designers. This analysis was made possible through the availability of chat transcripts that documented daily interactions in real time.

Approaching design at the interaction level of analysis can provide deeper insight into understanding of the relationship between attention, action and product, without post hoc rationalization that is common in interviews. Attention has often been measured through text analysis of letters to shareholders or annual reports (e.g. Cho and Hambrick 2006), but these indirect methods reflect self-serving and retrospective reporting biases (Barr 1998; Osborne et al. 2001; Yu, Engleman & Van de Ven, 2005). Use of chat transcripts as a data source thus allows for analysis into the messy (Zahra, 2007) and nonlinear design process in the entrepreneurial context.

6.4.4 Limitations of the Digital Exoskeleton

Lastly, I elaborate on my conceptualization of the digital exoskeleton construct. Different constructs were compared and contrasted when trying to label the resulting design product. The software was not created to support a firm's operations, as in the digital façade (Tumbas et al., 2015) but instead was the core of the digital business. The design product was thus constitutive of the business and designer activities that shaped the digital business and the emergence of the eventual software. Constructs, such as scaffolds (Orlikowski, 2006) and facades (Tumbas et al. 2015), were explored in my search for a construct that captured the essence of the design product.

These scaffold and digital façade concepts suggest that the function of the software in the early years was a temporary support used in the design of the eventual software. It is important to note that what HealthCom was designing is not a support but rather a working version of the software that was licensed to clients for use in a high-reliability context. None of the existing constructs captured the function of the design product as a working product that is in use, but yet constantly evolving.

The digital exoskeleton introduces risks that would be particularly salient for clients in high-reliability settings. While some of the characteristics of digital exoskeletons has been acknowledged among the practitioners, a design product, like the digital exoskeleton, has received limited attention in academic research (Rakitin, 2001, 2005). The nature of the digital exoskeleton goes again software design principle of software quality and user centered design processes, as opposed to disposable code that is thrown together with "little or no respect for engineering discipline" (Vidgen & Wang, 2009).

It is important to note that my current conceptualization of the digital exoskeleton is relevant only to startups in their early years. During this stage, designers are still searching for the appropriate product market fit. The designers feel most strongly the tension between financial-focused designs and software quality-focused design. Consequently, they adopt the design actions highlighted in this dissertation, which culminates in the digital exoskeleton. Beyond this stage, funding received is used to design more robust software (Delventhal, 2017) The designers would have the resources to adopt the software design methodologies reviewed in Chapter Two and be more user centered, or attentive to the quality-focused attention drivers, depending on the organization's stage of life (Adizes, 1979).

6.5 IMPLICATIONS FOR PRACTICE

6.5.1 Implications for Designers

Designers in early startups need to recognize the limitations of espoused software design methods and to identify the relevance of these practices for their design process and context. The ability of the digital exoskeleton to continue to help the startup meet revenue targets may be short lived. Vulnerability in the digital exoskeleton may eventually inhibit the startup's exploration and exploitation of the market opportunities. Thus, designers

should recognize the need to balance the benefits of the postponing investment in design with its costs over time.

6.5.2 Implications for Scaling

Attention needs to be allocated to the pursuit of these quality related designs, or else the quality of the design product may atrophy. Use of flexible design methods, such as Agile, has been found to compromise quality over time, as a result of inadequate architecture planning, low levels of test coverage (Highsmith & Cockburn, 2001; Boehm, 2002). Attention away from quality leads to problems with architectural scalability, which prevents the achievement of agility (Conboy, 2009).

The goal of the design actions outlined in this dissertation was to generate new features for the capturing of business opportunities. The quality of the design or architecture is under-prioritized, as observed by the use of coupling and postponing in the design. These design actions exacerbate the intensity of technical debt beyond that discussed in traditional software design or Agile design methods (Elbana & Sarker, 2016). The shortened timeline and ambiguity in quality further increase the designers' ability to adopt quality compromises, as quality is difficult to evaluate in novel software. Repayment of technical debt is also postponed until resources became available in the next round of funding. Specifically, the Series A and B funding rounds provide funding that are meant for the design of robust and scalable software. However, postponement of attention allocation to architecture and coupling will lead to difficulty scaling in the future. The digital exoskeleton is thus inappropriate for digital businesses with rapidly expanding user bases.

6.5.3 Implications for Investors

The findings can also inform investors about adapting to the rules of the game to include quality-related metrics for longer-term investments. The startup growth metrics suggest that investors view software design as secondary and as a vehicle for generating financial returns for the firm. However, they should recognize the impact of their rules of the game on the creation of the digital exoskeleton, and unsustainability of the digital exoskeleton over time, as it will inhibit the startup's ability to exploit opportunities with bigger clients.

6.6 Future Research

6.6.1 Relevance in Diverse Organizational Settings

Startups and entrepreneurs vary in their objectives, opportunity recognition and exploitation capabilities, and access to resources (Zahra et al., 2014). All of these factors can affect attention allocation in design, the design actions and design product. The dissertation has pointed to several contextual factors, such as stages of the firm, involvement of financially-oriented players, novel nature of software and ambiguity around quality. The dynamics may change in the design of a consumer software. Designing software for enterprises in a high reliability context with long negotiation cycles presents challenges for early stage startups that are in severe need of continued revenue. Further conceptual and empirical development can help disentangle the interactions between the contextual factors.

As researchers build on the findings in this dissertation to identify how attention of designers can vary or be directed, researchers can perhaps begin to identify ways in which the individual's actions and firm structure can mediate or moderate the occurrence of quality compromises in these settings. Startups in the early years might differ in terms of

the organizational culture or power relationships that might be different than those in the large companies. For instance, Adizies (1979) described how the founders might be the only management body involved in governing the firm during the early years. The decisions in startups could be made in a collaborative or hierarchical manner, which would shape how attention is allocated. In the case of HealthCom, the cofounders were observed to adopt a more hierarchical approach by limiting opportunities for communication within the firm about quality-focused issues. Researchers studying attention structures can also examine collaborative startups versus hierarchical decision-making to see if one is more likely to adopt financial-focused designs compared to the other. Consideration of the relationship and dynamics of power within startups and or even beyond the startup (e.g., dynamics between the investors and founders) can also shed light on attention allocation and the subsequent design actions undertaken.

6.6.2 Relevance across Stages of the Firm

The findings on the design actions and digital exoskeleton is relevant only for the early years of the startup's life. During the early years, the focus of the entrepreneur is on sales and involves constant experimentation (Adizies, 1979). Different attention structures will lead firms to create forms of digital exoskeletons with varying levels of these characteristics. One product design is not, ex ante, inherently superior to another, especially when evaluated by different players with distinct rules of the game.

Future studies could focus on the impact of financial-focused designs for firms in the mid- to long-term. Longitudinal methods are necessary to understand the changes over time, for instance, if financial-focused design becomes less practiced with resource acquisition. Future studies could take into consideration more fine-tuned controls, such as the nature of the digital business (e.g., consumer software versus enterprise software),

origins or absence of financial-focused attention drivers and whether they lead to variants of the digital exoskeleton.

6.6.3 Assessing the value of the Digital Exoskeleton

It is also difficult to disentangle the effect of the design actions on performance, which is a result of the interaction between the contextual factors and the actions of the designers. Future studies using methods, such as multiple case studies or simulation, can be used to assess the value of the digital exoskeleton to the startup's survival. At this moment, the ABV does not provide insight into the superiority or inferiority of the digital exoskeleton but, instead, highlights how the designers' actions, intentions, and attention vary according to the context.

The digital exoskeleton enabled HealthCom to exploit emerging business opportunities, which led to its eventual success (i.e., an acquisition). Conversely, if the digital exoskeleton led to a poor-quality design that inhibited contract negotiation with clients, this would have been a poor design product. The counterfactual could also occur. If the designers had paid attention to quality-focused designs, but the time taken to develop high quality designs led to missed opportunities, then the current investment in design will be viewed negatively. This dissertation sets the stage for future empirical research that looks beyond a single firm and opportunities for studying the financial-focused design approach over time and across firms.

6.7 Conclusion

The objective of this dissertation was to understand the impact of the entrepreneurial context on the design actions and design product in the early years. The findings suggest that the uncertainty, financial and time pressures led to design actions that

resulted in a digital exoskeleton. In a startup marked by high uncertainty, the design actions reflect how the designers allocated their attention and resources to manage the tensions. The design actions and digital exoskeleton highlighted depart from the recommendations of existing software design methods.

These influence of the entrepreneurial context on the design of a digital exoskeleton are not likely to dissipate anytime soon. In fact, as criteria for funding intensifies, it is likely that more extreme design actions and minimalist forms of digital exoskeleton may emerge. I hope that this study can raise additional questions and answers that contribute to an understanding of this burgeoning phenomenon.

References

Abdel-Hamid, T. K. (1989). A Study of Staff Turnover, Acquisition, and Assimilation and Their Impact on Software Development Cost and Schedule. *Journal of Management Information Systems*, *6*(1), 21–40.

Abrahamsson, P., Conboy, K., & Wang, X. (2009). "Lots done, more to do": the current state of agile systems development research. *European Journal of Information Systems*, *18*(4), 281–284.

Abrahamsson, P., Conboy, K., & Wang, X. (2009). Lots done, more to do: The current state of agile systems development research. *European Journal of Information Systems*, *18*(4), 281–284.

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods: {A} comparative analysis. *25th Int. Conference On Software Engineering, Proc.*, 244–254.

Adams, R. B., Licht, A. N., & Sagiv, L. (2011). Shareholders and stakeholder: How do directors decide? *Strategic Management Journal*, *32*(March), 1331–1355.

Adizes, I. (1979). Organizational passages—Diagnosing and treating lifecycle problems of organizations. *Organizational Dynamics*, *8*(1), 3–25.

Aldrich, H., Zimmer, C., & Jones, T. (1986). Small business still speak with the same voice: a replication of "the voice of small business and the politics of survival." *Sociological Review*, *34*(2), 335–356.

Alvarez, S. a., & Barney, J. B. (2013). Entrepreneurship : Comments on Venkataraman et al . (2012) and. *Academy of Management Review*, *38*(1), 154–160.

Anderson, E., Lim, S. Y., & Joglekar, N. (2017). Are More Frequent Releases Always Better? Dynamics of Pivoting, Scaling, and the Minimum Viable Product. *Hawaii International Conference on System Sciences 2017 (HICSS-50)*.

Anderson, P., & Tushman, M. L. (1990). Technological discontinuities and dominant designs: A cyclical model of technological change. *Administrative Science Quarterly,* 604-633.

Antonopoulou, K., Nandhakumar, J., & Henfridsson, O. (2016). Creating new value through repurposing digital innovations. In *AOM 2016* (pp. 1–40).

Antonopoulou, K., Nandhakumar, J., & Panourgias, N. S. (2014). Value proposition for digital technology innovations of uncertain market potential. *Twenty Second European Conference on Information Systems*, 1–16.

Apte, U., Sankar, C. S., Thakur, M., Bank, M., Turner, J. E., & Group, M. (1990). Reusability-Based Strategy for Development of Information Systems : Implementation Experience of a Bank Strategy, (December), 421–433.

Austin, R. D. (2001). The Effects of Time Pressure on Quality in Software Development: An Agency Model. *Information Systems Research*, *12*(2), 195–207.

Autio, E., Kenney, M., Mustar, P., Siegel, D., & Wright, M. (2014). Entrepreneurial innovation: The importance of context. *Research Policy*, *43*(7), 1097–1108.

Bailey, D. E., Leonardi, P. M., & Barley, S. R. (2012). The Lure of the Virtual.

Organization Science, *23*(5), 1485–1504.
Bamberger, P. (2008). From the Editors Beyond Contextualization: Using Context Theories to Narrow the Micro-Macro Gap in Management Research. *Academy of Management Journal*, *51*(5), 839–846.
Barnard, C. I. (1938). *The functions of the executive*. Cambridge, Massachussetts: Harvard University.
Barnett, M. (2008). An Attention-Based View of Real Options Reasoning. *The Academy of Management Review*, *33*(3), 606–628.
Barreto, I., & Patient, D. (2013). Toward a Theory of Intraorganizational Attention Based on Desirability and Feasibility Factors. *Strategic Management Journal*, *34*, 687–703.
Beck, K. (2003). *Test-driven development : by example*. Reading, MA: Addison-Wesley.
Beck, K. (2004). *Extreme Programming Explained*. Reading, MA: Addison-Wesley.
Berente, N., & Lyytinen, K. (2007). What Is Being Iterated? Reflections on Iteration in Information System Engineering Processes. In *Conceptual Modelling in Information Systems Engineering* (pp. 261–278). Berlin, Heidelberg: Springer Berlin Heidelberg.
Bharadwaj, A., Omar el sawy, Paul Pavlou, Venkatraman, N., El Sawy, O. a., Pavlou, P. a., & Venkatraman, N. (2013). Digital business strategy: Toward a next generation of insights. *MIS Quarterly*, *37*(2), 471–482.
Blank, S. (2013). Why the Lean Start Up Changes Everything. *Harvard Business Review*, *91*(5), 64.
Boehm, B., & Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, *22*(5), 30–39.
Cable, D. M., & Shane, S. (1997). A prisoner's dilemma approach to entrepreneur-venture capitalist relationships. *Academy of Management Review*, *22*(1), 142–176.
Campbell, K. (2015). Jennifer Chong From Linjer-From 0 to 10M in 3 years. Retrieved November 17, 2017, from https://www.huffingtonpost.com/entry/jennifer-chong-from-linjer-from-0-to-10m-in-3-years_us_59d195f9e4b0f58902e5cd93
Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems*, *18*(4), 332–343.
Cao, L., Ramesh, B., & Abdel-Hamid, T. (2010). Modeling dynamics in agile software development. *ACM Transactions on Management Information Systems*, *1*(1), 1–26.
Carmel, E., & Sawyer, S. (1998). Packaged software development teams: what makes them different? *Information Technology & People*, *11*(1), 7–19.
Chakraborty, S., Sarker, S. S., & Sarker, S. S. (2010a). An Exploration into the Process of Requirements Elicitation : A Grounded Approach. *Journal of the Association for Information Systems*, *11*(4), 212–249.
Chakraborty, S., Sarker, S., & Sarker, S. (2010b). An Exploration into the Process of Requirements Elicitation : A Grounded Approach. *Journal of the Association for Information Systems*, *11*(4), 212–249.
Charmaz, K., & Mitchell, R. G. (2001). Grounded theory in ethnography. Dans. In & L. L. PA Atkinson, A. Coffey, S. Delamont, J. Lofland (Ed.), *Handbook of ethnography* (pp. 160–174).

Chatterjee, S., Sarker, S., & Fuller, M. (2009). Ethical Information Systems Development : A Baumanian Postmodernist Perspective. *Journal of the Association for Information Systems*, *10*(11), 787–815.

Chen, W. R., & Miller, K. D. (2007). Situational and institutional determinants of firms' R&D search intensity. *Strategic Management Journal*, *28*(4), 368–381.

Cho, T. S., & Hambrick, D. C. (2006). Attention as the Mediator Between Top Management Team Characteristics and Strategic Change: The Case of Airline Deregulation. *Organization Science*, *17*(4), 453–469.

Collewaert, V., & Sapienza, H. J. (2016). How Does Angel Investor-Entrepreneur Conflict Affect Venture Innovation? It Depends. *Entrepreneurship: Theory and Practice*, *40*(3), 573–597.

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, *20*(3), 329–354.

Cooper, G., & Woolgar, S. (1994). Software quality as community performance. In *The Management of Information and Communication Technologies: emerging patterns of control*. London: Aslib.

Corbin, J., & Strauss, A. (2008). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Thousand Oaks.

Creswell, J. W., & Miller, D. L. (2000). Determining Validity in Qualitatife Inquiry. *Theory Into Practice*, *39*(3), 7.

David Ehrenberg. (2014). The Seven Startup Metrics You Must Track. Retrieved November 30, 2017, from https://www.forbes.com/sites/theyec/2014/06/20/the-seven-startup-metrics-you-must-track/#dcdde3725ec7

Davidsson, P. (2015). Entrepreneurial opportunities and the entrepreneurship nexus: A re-conceptualization. *Journal of Business Venturing*, *30*(5), 674–695.

Davis, J. P., Yin, P., & Davis, J. P. (2014). Experimentation Strategies and Entrepreneurial Innovation : Inherited Market Differences in the iPhone Ecosystem Market Differences in the iPhone Ecosystem.

Delventhal, S. (2017). Series A, B, C Funding: What It All Means and How It Works | Investopedia. Retrieved November 30, 2017, from https://www.investopedia.com/articles/personal-finance/102015/series-b-c-funding-what-it-all-means-and-how-it-works.asp

DeMarco, T. (1995). *Why does software cost so much? : and other puzzles of the information age*. Dorset House Publishing Co., Inc.

Dutton, J. E., & Dukerich, J. M. (1991). Keeping an Eye on the Mirror: Image and Identity in Organizational Adaptation. *Academy of Management Journal*, *34*(3), 517–554.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9–10), 833–859.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review*, *14*(4), 532–550.

Eisenhardt, K. M., & Martin, J. a. (2000). Dynamic Capabilities: What Are They? *The SMS Blackwell Handbook of Organizational Capabilities*, *21*(10), 341–363.

Eisenmann, T. R., Ries, E., & Dillard, S. (2012). *Hypothesis-Driven Entrepreneurship:*

The Lean Startup.
Elbanna, A., & Sarker, S. (2016). The Risks of Agile Software Development: Learning from Adopters. *IEEE Software*, *33*(5), 72–79.
Emerson, R. M., Fretz, R. I., & Shaw, L. L. (2001). Participant observation and fieldnotes. In *Handbook of Ethnography* (pp. 352–368).
Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research. *Journal of Database Management*, *16*(4), 88–100.
Exoskeleton. (n.d.). Retrieved November 3, 2017, from https://en.wikipedia.org/wiki/Exoskeleton
Faraj, S., & Sproull, L. (2000). Coordinating Expertise in Software Development Teams. *Management Science*, *46*(12), 1554–1568.
Fetterman, D. (1998). *Ethnography.*
Fiet, J. O. (1995). Risk avoidance strategies in venture capital markets. *Journal of Management Studies*, *32*(4), 551–574.
Fiol, C. M., & O'Connor, E. J. (2003). Waking Up! Mindfulness in the Face of Bandwagons. *Academy of Management Review*, *28*(1), 54–70.
Fitzgerald, B. (1997). The use of systems development methodologies in practice: a field study. *Information Systems Journal*, *7*(3), 201–212.
Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems*, *15*(2), 197–210.
Gallivan, M. J., & Keil, M. (2003). a critical case study, 37–68.
Garud, R., Jain, S., & Tuertscher, P. (2008). Incomplete by Design and Designing for Incompleteness. *Organization Studies*, *29*(3), 351–371.
George, G. (2005). Slack resources and the performance of privately held firms. *Academy of Management Journal*, *48*(4), 661–676. https://doi.org/10.5465/AMJ.2005.17843944
Gersick, C. J. G. (1994). Pacing strategic change: the case of a new venture. *Academy of Management Journal*, *37*(1), 9–45.
Gimmon, E., & Levie, J. (2010). Founder's human capital, external investment, and the survival of new high-technology ventures. *Research Policy*, *39*(9), 1214–1226.
Glaser, B. G. (1965). The Constant Comparative Method of Qualitative Analysis. *Social Problems*, *12*(4), 436–445.
Glaser, B. G. (1978). Theoretical sensitivity: Advances in the methodology of grounded theory.
Gomez-Mejia, L. R., Balkin, D. B., & Welbourne, T. M. (1990). Influence of venture capitalists on high tech management. *The Journal of High Technology Management Research*, *1*(1), 103–118.
Grilli, L., & Murtinu, S. (2012). Government , venture capital and the growth of European high-tech entrepreneurial firms, *43*(217485), 1–55.
Hansen, S., & Lyytinen, K. (2010). Challenges in contemporary requirements practice. *Proceedings of the 43rd Hawaii International Conference on System Sciences*, 1–11.
Hanseth, O., & Lyytinen, K. (2010). Design theory for dynamic complexity in information

infrastructures: The case of building internet. *Journal of Information Technology*, *25*(1), 1–19.
Harris, M. L., Collins, R. W., & Hevner, A. R. (2009). Control of flexible software development under uncertainty. *Information Systems Research*, *20*(3), 400–419.
He, J., & King, W. R. (2008). The Role of User Participation in Information Systems Development : Implications from a Meta- Analysis The Role of User Participation in Information Systems Development : Implications from a Meta-Analysis, *1222*(December).
Henfridsson, O., Yoo, Y., & Svahn, F. (2009). Path creation in digital innovation: A multi-layered dialectics perspective. *Working Papers on Information Systems*, *9*(2009), 1–26.
Highsmith, J. (2002). What Is Agile Software Development? *The Journal of Defense Software Engineering*, *15*(10), 4–9.
Highsmith, J. A. (2002). *Agile software development ecosystems*. Addison-Wesley Professional.
Highsmith, J., & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer*, *34*(9), 120–127.
Hoffman, A. J., & Ocasio, W. (2001). Not All Events Are Attended Equally: Toward a Middle-Range Theory of Industry Attention to External Events. *Organization Science*, *12*(4), 414–434.
Huang, M.-Hu., Rand, W., & Rust, R. T. (2016). Don't Do It Right, Do It Fast? Speed and Quality of Innovation as an Emergent Process. *International Conference on Information Systems*, 1–19.
Hylving, L., Henfridsson, O., & Selander, L. (2012). The Role of Dominant Design in a Product Developing Firm's Digital Innovation. *Journal of Information Technology and Application*, *13*(2), 5–21.
Iansiti, M., & Gill, G. (1990). Microsoft corporation: Office business unit. *Harvard Business School Case*, *9–691–033,*.
Iivari, J., & Iivari, N. (2006). Varieties of User-Centeredness. *Analysis*, *0*(C), 1–10.
Iivari, J., & Maansaari, J. (1998). The usage of systems development methods: Are we stuck to old practices? *Information and Software Technology*, *40*(9), 501–510.
Jain, B. a, & Kini, O. (1995). Venture capitalist participation and the post issue operating performance of IPO firms. *Managerial and Decision Economics*, *16*(1989), 593–606.
Johns, G. (2006). The essential impact of context on organizational behavior. *Academy of Management Review*, *31*(2), 386–408.
Kaplan, S. (2008). Framing Contests: Strategy Making Under Uncertainty. *Organization Science*, *19*(5), 729–752.
Keil, M., & Carmel, E. (1995). Customer-Developer Links in Software Development. *Communications of the ACM*, *38*(5), 33–44.
Kranz, J. J., Hanelt, A., & Kolbe, L. M. (2016). Understanding the influence of absorptive capacity and ambidexterity on the process of business model change - the case of on-premise and cloud-computing software. *Information Systems Journal*, n/a-n/a.
Kujala, S., & Väänänen-Vainio-Mattila, K. (2008). Value of Information Systems and

Products: Understanding the Users' Perspective and Values. *JITTA: Journal of Information Technology Theory and Application*, *9*(4), 23.
Kvale, S., & Brinkmann, S. (2009). *Learning the craft of qualitative research interviewing*. Thousands Oaks: Sage Publications.
Kyprianou, C., Graebner, M. E., & Rindova, V. (2015). Strategic Conversations. In *Handbook of Qualitative Organizational Research: Innovative Pathways and Methods* (p. 272).
LeCompte, M. D., & Goetz, J. P. (1982). Problems of Reliability and Validity in Ethnographic Research. *Review of Educational Research*, *52*(1), 31–60.
Lee, G., & Xia, W. (2010). Toward Agile: an Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility. *MIS Quarterly*, *34*(1), 87–114.
Lehmann, J., & Rosenkranz, C. (2017). The Trajectories of Digital Entrepreneurship : Disentangling the Digital, (Stewart), 1–13.
Leong, C., Pan, S. L., Newell, S., & Cui, L. (2016). the Emergence of Self-Organizing E-Commerce Ecosystems in Remote Villages of China: a Tale of Digital Empowerment for Rural Development. *MIS Quarterly*, *40*(2), 475-A8.
Levie, J., Autio, E., Acs, Z., & Hart, M. (2014). Global entrepreneurship and institutions: An introduction. *Small Business Economics*, *42*(3), 437–444.
Levina, N., & Vaast, E. (2016). Leveraging archival data from online communities for grounded process theorizing. In *Handbook of Qualitative Organizational Research: Innovative Pathways and Methods* (pp. 215–224).
Lim, S. Y., & Anderson, E. G. (2016). Institutional Barriers Against Innovation Diffusion: From the Perspective of Digital Health Startups. In *2016 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 3328–3337). IEEE.
Lim, S. Y., Jarvenpaa, S. L., & Lanham, H. J. (2015). Barriers to Interorganizational Knowledge Transfer in Post-Hospital Care Transitions: Review and Directions for Information Systems Research. *Journal of Management Information Systems*, *32*(3), 48–74.
Lyytinen, K. (1987). Different Perspectives on Information Systems: Problems and Solutions. *ACM Computing Surveys*, *19*(1), 1–42.
Lyytinen, K. (1988). Expectation failure concept and systems analysts' view of information system failures: Results of an exploratory study. *Information & Management*, *14*(1), 45–56.
Lyytinen, K., & Newman, M. (2015). A tale of two coalitions - marginalising the users while successfully implementing an enterprise resource planning system. *Information Systems Journal*, *25*(2), 71–101.
Lyytinen, K., & Robey, D. (1999). Learning failure in information systems development. *Information Systems Journal*, *9*(2), 85–101.
Majchrzak, A., & Beath, C. (2001). Beyond user participation: A process model of learning and negotiation during systems development. *Redefining the Organizational Roles of Information Technology in the Information Age*.
Marakas, G. M., & Elam, J. J. (1998). Semantic structuring in analyst acquisition and

representation of facts in requirements analysis. *Information Systems Research*, *9*(1), 37–63.
Marakas, G. M., Yi, M. Y., Johnson, R. D., & Introduction, I. (1998). The Multilevel and Multifaceted Character of Computer Self-Efficacy : Toward Clarification of the Construct and an Integrative Framework for Research, 126–163.
March, J. G., & Olsen, J. P. (1975). The uncertainty of the past: organizational learning under ambiguity. *European Journal of Political Research*, *3*(2), 147–171.
March, J. G., & Simon, H. . (1958). *Organizations*. Oxford, England: Wiley.
March, J., & Shapira, Z. (1992). Variable Risk Preferences and the Focus of Attention. *Psychological Review*, *99*(1), 172–183.
Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research*, *20*(3), 377–399.
Maruping, L. M., Zhang, X., & Venkatesh, V. (2009). Role of collective ownership and coding standards in coordinating expertise in software project teams. *European Journal of Information Systems*, *18*(4), 355–371.
Mathias, B. D., & Williams, D. W. (2014). The Impact of Role Identities on Entrepreneurs' Evaluation and Selection of Opportunities. *Journal of Management*, *43*(3), 892–918.
Mathiassen, L., Seewaldt, T., & Stage, J. (1995). Prototyping and Specifying: Principles and Practices of a Mixed Approach. *Scandinavian Journal of Information Systems*
McMullen, J. S., Shepherd, D. A., & Patzelt, H. (2009). Managerial (In)attention to competitive threats. *Journal of Management Studies*, *46*(2), 157–181.
Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook.* Sage.
Molnar, W., & Nandhakumar, J. (2009). Managing projects in an embedded system development context: An in-depth case study from an improvisational perspective. *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, 1–10.
Nambisan, S. (2016). Digital Entrepreneurship: Toward a Digital Technology Perspective of Entrepreneurship. *Entrepreneurship Theory and Practice2*, (October 2016).
Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72–78.
Nidumolu, S. R., & Subramani, M. R. (2003). The Matrix of Control: Combining Process and Structure Approaches to Managing Software Development. *Journal of Management Information Systems*, *20*(3), 159–196.
Ocasio, W. (1997). Towards an Attention Based View of the Firm. *Strategic Management Journal*, *18*(S1), 187–206.
Ocasio, W., & Joseph, J. (2008). Rise and Fall - or Transformation?. The Evolution of Strategic Planning at the General Electric Company, 1940-2006. *Long Range Planning*, *41*(3), 248–272.
Ojala, A. (2015). Business models and opportunity creation: How IT entrepreneurs create and develop business models under uncertainty. *Information Systems Journal*, *25*(5), 451–476.

Onyemah, V., Pesquera, M. R., & Ali, A. (2013). What entrepreneurs get wrong. *Harvard Business Review*, *91*(5).
Orlikowski, W. J., & Iacono, C. S. (2001). Research — A Call to Theorizing the IT Artifact Research Commentary : Desperately Seeking the " IT " in IT Research — A Call to Theorizing the IT Artifact, (January 2014).
Park, J., Boland, R., & Yoo, Y. (2011). Discovering the Meanings of Design in IS: Reviews and Future Directions (pp. 124–137). Springer, Berlin, Heidelberg.
Parnas, D. L., & Clements, P. C. (1986). A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, *12*(2), 251–257.
Persson, J. S., Mathiassen, L., & Aaen, I. (2012). Agile distributed software development: Enacting control through media and context. *Information Systems Journal*, *22*(6), 411–433.
Politis, D. (2008). Business angels and value added what do we know and where do we go. *Venture*, *10*(2), 127–147.
Port, D., & Bui, T. (2009). Simulating mixed agile and plan-based requirements prioritization strategies: Proof-of-concept and practical implications. *European Journal of Information Systems*, *18*(4), 317–331.
Relander, B. (2015). N Number One Country For Tech Start-Ups: U.S.A. Retrieved December 5, 2017, from https://www.investopedia.com/articles/investing/022815/1-country-tech-startups-usa.asp
Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses.* (R. H. LLC, Ed.).
Rumelt, R. P. (1984). Towards a Strategic Theory of the Firm. Competitive strategic management. *Competitive Strategic Management*, 556–570.
Santos, F. M., & Eisenhardt, K. M. (2009). Constructing Markets and Shaping Boundaries: Entrepreneurial Power in Nascent Fields. *Academy of Management Journal*, *52*(4), 643–671.
Sapienza, H. J., De Clercq, D., & Sandberg, W. R. (2005). Antecedents of international and domestic learning effort. *Journal of Business Venturing*, *20*(4), 437–457.
Sapienza, H. J., Korsgaard, M. A., Academy, T., & Jun, N. (2007). Procedural Justice in Entrepreneur-Investor Relations, *39*(3), 544–574.
Sawyer, S. (2001a). Effects of intra-group conflict on packaged software development team performance. *Information Systems Journal*, *11*(2), 155–178.
Sawyer, S. (2001b). Software Development Team Performance. *Information Systems Journal*, *XY*(IS), 155–178. Retrieved from
Schonberger, B. R. J. (2010). MIS design : A contingency approach, *4*(1), 13–20.
Schoonhoven, C., Eisenhardt, K. M., & Lyman, K. (1990). Speeding Products to Market: Waiting Time to First Product. *Administrative Science Quarterly*, *35*(1), 177–207.
Selznick, P. (1957). *Leadership in administration: A sociological interpretation*. Berkeley. California.
Shane, S., & Cable, D. (2002). Network Ties, Reputation, and the Financing of New Ventures. *Management Science*, *48*(3), 364–381.
Sharkey, M. (2013). 6 things wrong with the "Lean Startup" model (and what to do about

it). Retrieved from https://venturebeat.com/2013/10/16/lean-startups-boo/
Shepherd, D. A., Mcmullen, J. S., & Ocasio, W. (2017). Is that an opportunity? An attention model of top managers' opportunity beliefs for strategic action. *Strategic Management Journal*, *38*(3), 626–644.
Shepherd, D. A., & Zacharakis, A. (2001). Speed to Initial Public Offering of VC-Backed Companies. *Entrepreneurship: Theory & Practice*, *25*(3), 59.
Shimizu, K. (2007). Prospect Theory , Behavioral Theory , and the Threat - Rigidity Thesis : Combinative Effects on Organizational Decisions to Divest Fomerly Acquired Units Problemstellung Grundidee Hypothesen. *Academy of Management Journal*, *50*(6), 1495–1514.
Siddiqi, J., & Shekaran, C. (1996). Requirement engineeting: the emerging wisdom. *IEEE*, 15–19.
Simon, H. A. (1947). *Administrative behavior*. New York, NY: Macmillan.
Sommerville, I. (1996). Software process models. *ACM Computing Surveys*, *28*(1), 269–271.
Spradley, J. P. (1979). *The ethnographic interview*. New York.
Stacey, P., & Nandhakumar, J. (2006). Responding to games development challenges through mood-mediated improvisation. *Ecis*, (2006).
Starbuck, W. H., & Milliken, F. J. (1988). Executive perceptual filters: What they notice and how they make sense. *The Executive Effect: Concepts and Methods for Studying Top Managers*, (June 1975), 35–65.
Steier, L., & Greenwood, R. (1995). Venture capitalist relationships in the deal structuring and post-investment stages of new firm creation. *Journal of Management Studies*, *32*(3), 337–357.
Stevens, R., Moray, N., Bruneel, J., & Clarysse, B. (2015). Attention allocation to multiple goals: The case of for-profit social enterprises. In *Strategic Management Journal* (Vol. 36, pp. 1006–1016).
Strauss, A., & Corbin, J. (1998). *Basics of qualitative research techniques*.
Sutton, R. I., & Hargadon, A. (1996). Brainstorming Groups in Conext: Effectiveness in a Product Design Firm. *Administrative Science Quarterly*.
Sydow, J., Schreyögg, G., & Koch, J. (2009). Organizational Path Dependence: Opening the Black Box. *Academy of Management Review*, *34*(4), 689–709.
Thanasankit, T. (2002). Requirements engineering—exploring the influence of power and Thai values. *European Journal of Information Systems*, *11*(2), 128–141.
Thomas, L. D. W., & Autio, E. (2012). Modeling the ecosystem: A meta-synthesis of ecosystem and related literatures. *The DRUID Society Conference on Innovation and Competitiveness - Dynamics of Organizations, Industries, Systems and Regions*, 0–27.
Tuggle, C. S., Schnatterly, K., & Johnson, R. A. (2010). Attention patterns in the boardroom: How board composition and processes affect discussion of entrepreneurial issues. *Academy of Management Journal*, *53*(3), 550–571.
Tumbas, S., Berente, N., & vom Brocke, J. (2017). Born Digital: Growth Trajectories of Entrepreneurial Organizations Spanning Institutional Fields. *ICIS 2017 Proceedings*,

1–20.
Tumbas, S., Seidel, S., Berente, N., & Brocke, J. (2015). The " Digital Façade " of Rapidly Growing Entrepreneurial Organizations. *Proceedings of the 36th International Conference on Information Systems (ICIS)*, (July 2016), 1–19.
Utterback, J. M., & Abernathy, W. (2003). A Dynamic Model of Process and Product Innovation, 1–18.
Van de Ven AH, Polley DE, Garud R, V. S. (1999). *The innovation journey*. New York, NY: Oxford University Press.
Van Doorn, S., Jansen, J. J. P., Van Den Bosch, F. A. J., & Volberda, H. W. (2013). Entrepreneurial orientation and firm performance: Drawing attention to the senior team. *Journal of Product Innovation Management*, *30*(5), 821–836.
Vidgen, R., & Wang, X. (2009). Coevolving Systems and the Organization of Agile Software Development. *Information Systems Research*, *20*(3), 355–376.
Vohora, A., Wright, M., & Lockett, A. (2004). Critical junctures in the development of university high-tech spinout companies. *Research Policy*, *33*(1), 147–175.
Vuori, T. O., & Huy, Q. N. (2016). Distributed Attention and Shared Emotions in the Innovation Process: How Nokia Lost the Smartphone Battle. *Administrative Science Quarterly*, *61*(1), 9–51.
Walsham, G. (1995). Interpretive case studies in IS research- nature and method. *European Journal of Information Systems*, *4*, 74–81. Retrieved from
Yin, R. (1994). *Case study research: Design and methods*. Beverly Hills.
Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). Research Commentary: The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research *Information Systems Research 14*(4), 724–735.
Zahra, S. A. (2007). Contextualizing theory building in entrepreneurship research. *Journal of Business Venturing*, *22*(3), 443–452.
Zahra, S. A., Wright, M., & Abdelgawad, S. G. (2014). Contextualization and the advancement of entrepreneurship research. *International Small Business Journal*, *32*(5), 479–500.
Zmud, R. W. (1980). Management of large software development efforts. *MIS Quarterly: Management Information Systems*, *4*(2), 45–55.